2013 IEEE 13th International Conference on Data Mining

On the Feature Discovery for App Usage Prediction in Smartphones

Zhung-Xun Liao*, Shou-Chung Li*, Wen-Chih Peng*, Philip S Yu† and Te-Chuan Liu‡
*Department of Computer Science,
National Chiao Tung University, HsinChu, Taiwan
Email: {zxliao.cs96g, scli.cs02g, wcpeng}@nctu.edu.tw
†Department of Computer Science,
University of Illinois at Chicago, Chicago, IL, USA
Email: psyu@cs.uic.edu
‡Department of Studio,
HTC Corporation, Taipei, Taiwan
Email: philip_liu@htc.com

***Abstract*—With the increasing number of mobile Apps developed, they are now closely integrated into daily life. In this paper, we develop a framework to predict mobile Apps that are most likely to be used regarding the current device status of a smartphone. Such an Apps usage prediction framework is a crucial prerequisite for fast App launching, intelligent user experience, and power management of smartphones. By analyzing real App usage log data, we discover two kinds of features: The Explicit Feature (EF) from sensing readings of built-in sensors, and the Implicit Feature (IF) from App usage relations. The IF feature is derived by constructing the proposed App Usage Graph (abbreviated as AUG) that models App usage transitions. In light of AUG, we are able to discover usage relations among Apps. Since users may have different usage behaviors on their smartphones, we further propose one personalized feature selection algorithm. We explore minimum description length (MDL) from the training data and select those features which need less length to describe the training data. The personalized feature selection can successfully reduce the log size and the prediction time. Finally, we adopt the kNN classification model to predict Apps usage. Note that through the features selected by the proposed personalized feature selection algorithm, we only need to keep these features, which in turn reduces the prediction time and avoids the curse of dimensionality when using the kNN classifier. The results based on a real dataset demonstrate the effectiveness of the proposed framework and show the predictive capability for App usage prediction.**

***Keywords*-Mobile Application; Usage Prediction; Classification; Apps;**

I. INTRODUCTION

With the increasing number of smartphones, mobile applications (Apps) have been developed rapidly to satisfy users' needs [1], [2], [3], [4]. Users can easily download and install Apps on their smartphones to facilitate their daily lives. By analyzing the collected real Apps usage log data, the average number of Apps in a user's smartphone is around 56. For some users, the number of Apps is up to 150. As many Apps are installed on a smartphone, users need to spend more time swiping screens and finding the Apps they want to use. From our observation, each user has on average 40 launches per day. In addition, the launch delay of Apps becomes longer as their functionality becomes more complicated. To ease the inconvenience of searching for Apps [5], [6] and to reduce the delay in launching Apps [7], one possible way is to predict which Apps will be used before the user actually needs them.

Recently, some research works have addressed the Apps usage prediction problems [5], [6], [7]. In [5], only the temporal information of App usages is considered. However, some Apps with no significant periods cannot be predicted by their temporal profiles. In [7], the authors adopted three features to remedy slow App launches: time, location, and used Apps. However, they always use these three features to predict different users' usage, which is impractical as users could have different usage behavior. Though the authors in [6] collected 37 sensor reading as features, it cannot filter out useless features in advance, such that the system need to waste the storage and energy to process those useless features.

In this paper, we adopt the concept of minimum description length (MDL) to select personalized features for different users and propose a kNN-based App Prediction framework, called KAP, to predict Apps usage. Once we distinguish the useful and useless features, only the useful features need to be collected. Therefore, the size of the log data could be reduced. The overall framework is shown in Figure 1. KAP investigates features from both explicit and implicit aspects. The explicit feature is a set of sensor readings from built-in hardware sensors, such as GPS, time, accelerometers, etc. On the other hand, the implicit feature is referred to as the correlations of Apps usage. To capture these correlations, the implicit feature is represented as the transition probability among Apps.

The major contributions of this research work are summarized as follows.

- We address the problem of Apps usage prediction by discovering different feature sets to fulfill different users' Apps usage behavior, and propose the concept of

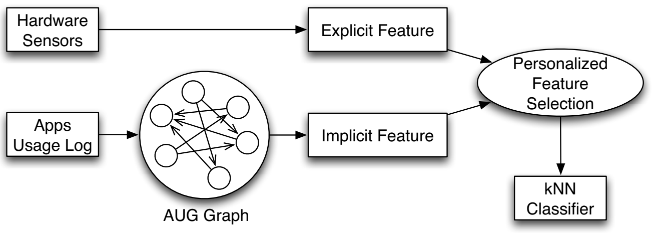


Figure 1. Overview of kNN-based App Prediction framework.

explicit and implicit features for Apps usage prediction.

- We estimate the distribution of the transition probability among Apps and design an Apps Usage Graph (AUG) to model both Apps usage order and transition intervals. Two algorithms are proposed to extract the implicit features from the AUG graph for training and testing purposes respectively.
- We propose a personalized feature selection algorithm in which one could explore MDL to determine a personalized set of features while still guaranteeing the accuracy of the predictions.
- A performance evaluation is conducted on real datasets, and our proposed framework outperforms the state-of-the-art methods [6].

The rest of this paper is organized as follows. Section II investigates the related works which discuss the conventional prediction problem and Apps usage prediction. Section III introduces the explicit and implicit features. Section IV presents the mechanism of personalized feature selection. Section V conducts extensive and comprehensive experiments. Finally, this paper is concluded with Section VI.

II. RELATED WORKS

To the best of our knowledge, the prediction problem of Apps usage in this paper is quite different from the conventional works. We focus on not only analysing usage history to model users' behavior, but on personalizing varied types of features including hardware and software sensors attached to smartphones. The proposed algorithm selects different features for different users to satisfy their usage behavior. Although there have been many research works solving the prediction problem in different domains [8], [9], [10], [11]. In [12], the author selected features from multiple data streams, but the goal is to solve the communication problem in a distributed system.

Currently, only a few studies discuss mobile Apps usage prediction. In [7], the authors solved the prediction problem through multiple features from 1) location, 2) temporal burst, and 3) trigger/follower relation. However, they did not analyze the importance of each feature. Therefore, for different users, they always use the same three features to predict their Apps usage. In [6], the authors investigated all possible sensors attached to a smartphone and adopted a Naive Bayes classification to predict the Apps usage. However, collecting all possible sensors is inefficient and impractical. Moreover, the useful sensors for different users could vary according to users' usage behavior. We claim that for different users, we need to use different sets of features to predict their usage. In this paper, we collect only the subset of all features which are personalized for different users.

III. EXPLICIT AND IMPLICIT FEATURES

In this paper, we separate the features into two main categories: the explicit feature and the implicit feature. The explicit feature represents the sensor readings which are explicitly readable and observable. The implicit feature is the Apps usage relations.

A. Explicit Feature Collection

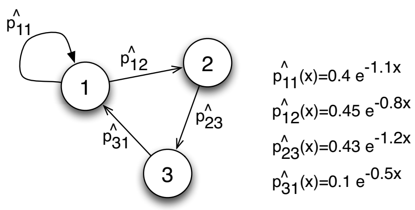
The number of hardware sensors we use for the explicit feature is 24, and we categorize these sensors on a smartphone into three groups: 1) device sensors, which sense the status inside the devices, for instance, space, free ram, and battery level; 2) environmental sensors, which capture the outside status such as time, GSM signal, and Wi-Fi signal; 3) personal sensors, which capture the outside status of the devices, for example, the acceleration, speed, heading and position of the device. It is totally free to add or remove any hardware sensors here.

B. Implicit Feature Extraction

The implicit feature formulates the usage transitions among Apps in a usage session. As mentioned in [7], users use a series of Apps, called a usage session, to complete a specific task. For example, one user could use "Maps" when travelling to a sightseeing spot, then use camera to take photos, and upload those photos to Facebook. Thus, the series of using "Maps", "Camera" and "Facebook" is called a usage session, denoted as "Map"$\xrightarrow{\delta_1}$"Camera"$\xrightarrow{\delta_2}$"Facebook", where δ_1 and δ_2 represent the transition intervals.

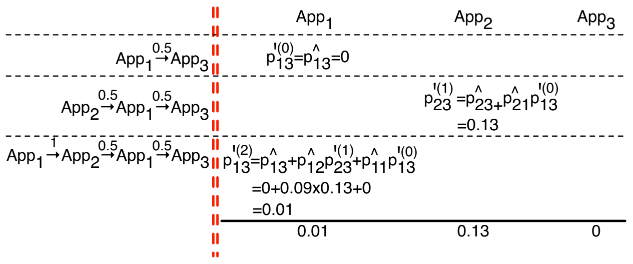
The implicit feature of "Facebook" in this usage session is thus $< s_{MF}(\delta_1), s_{CF}(\delta_1+\delta_2), s_{FF}(\infty) >$, where $s_{MF}(\cdot)$, $s_{CF}(\cdot)$, and $s_{FF}(\cdot)$ are probability models which represent the probability of using "Maps", "Camera" and "Facebook" respectively before using "Facebook" with the transition interval as the random variable. Note that because there is no "Facebook" to "Facebook" in this usage session, the transition interval is thus set to ∞ and then the probability would be 0.

1) Apps Usage Graph (AUG): For each user, we construct an Apps Usage Graph (AUG) to describe the transition probability among Apps. An AUG is a directed graph where each node is an App, the direction of an edge between two nodes represents the usage order, and the weight on each edge is a probability distribution of the interval between two Apps. Since two consecutive launches could be viewed



$\hat{p}_{11}$
$\hat{p}_{12}$
$\hat{p}_{23}$
$\hat{p}_{31}$
1
2
3
$\hat{p}_{11}(x)=0.4\ e^{-1.1x}$
$\hat{p}_{12}(x)=0.45\ e^{-0.8x}$
$\hat{p}_{23}(x)=0.43\ e^{-1.2x}$
$\hat{p}_{31}(x)=0.1\ e^{-0.5x}$

Figure 2. An example of the Apps Usage Graph (AUG).



App_1
App_2
App_3
$App_1 \xrightarrow{0.5} App_3$
$p'^{(0)}_{13}=\hat{p}_{13}=0$
$App_2 \xrightarrow{0.5} App_1 \xrightarrow{0.5} App_3$
$p'^{(1)}_{23}=\hat{p}_{23}+\hat{p}_{21}p'^{(0)}_{13}$
$=0.13$
$App_1 \xrightarrow{1} App_2 \xrightarrow{0.5} App_1 \xrightarrow{0.5} App_3$
$p'^{(2)}_{13}=\hat{p}_{13}+\hat{p}_{12}p'^{(1)}_{23}+\hat{p}_{11}p'^{(0)}_{13}$
$=0+0.09\text{x}0.13+0$
$=0.01$
0.01
0.13
0

Figure 3. Steps of obtaining the implicit feature of App_3 in the training case, $\cdots \rightarrow App_1 \xrightarrow{1} App_2 \xrightarrow{0.5} App_1 \xrightarrow{0.5} App_3$.

as a Poisson arrival process, we can formulate the intervals between two launches as an exponential distribution.

Here, Equation 1 formulates the exponential density function of the launch interval being in $[x, x+1)$. The parameter $\alpha = \hat{p(0)}$ is derived by assigning $x = 0$ in Equation 1, and could be calculated by $p(0)$, the real probability derived from the training data. Then, β is solved by minimizing the difference between the estimated probability $\hat{p(i)}$ and the real probability $p(i)$ as shown in Equation 2 for every interval i.

$$\hat{p(x)} = \alpha \exp^{-\beta x} \tag{1}$$

$$\begin{aligned} \beta &= \underset{\beta}{\operatorname{argmin}} \sum_i |\hat{p(i)} - p(i)| \\ &= \underset{\beta}{\operatorname{argmin}} \sum_i |p(0) \exp^{-\beta i} - p(i)| \end{aligned} \tag{2}$$

For example, Figure 2 shows an AUG with three Apps. From Figure 2, the probability of two consecutive usages of App_1 with an interval of 0.3 minutes (i.e., $App_1 \xrightarrow{0.3} App_1$) is 0.4, and $App_1 \xrightarrow{1.5} App_2$ is 0.2. Although AUG only takes two consecutive Apps into account, such as p_{12} and p_{23}, the probability of p_{13}, could be calculated by $p_{12} \times p_{23}$.

2) Implicit Features for Training: For each training case, the implicit features are derived by looking up the AUG. Suppose the currently used App (i.e., class label) is App_t, the implicit feature is thus, $< p'_{1t}, p'_{2t}, ..., p'_{nt} >$, where p'_{it} represents the probability of transiting from App_i to any random Apps and then to App_t. The probability of $p'^{(l)}_{it}$ is defined as in Equation 3 which is the summation of every probability from App_i to App_t. Note that we use a superscript, s, to indicate how many Apps are between App_i and App_t, and App_{m_k} is the k-th App after App_i. Once we derive the implicit feature in a reverse time order, the sub-problem of estimating $p'^{(s-k)}_{m_k,t}$ is already solved. The calculation of the implicit feature for App_i stops when the transition probability falls below a given threshold, min_{tp}. In our collected dataset, the transition probability falls to 0.1% when we look backward to more than 5 Apps, which is the default parameter for min_{tp}. Algorithm 1 depicts the derivation of the implicit feature for a training case with App_t as its class label.

$$p'^{(l)}_{it} = \hat{p_{it}} + \sum_k \hat{p_{i,m_k}} \times p'^{(s-k)}_{m_k,t} \tag{3}$$

Algorithm 1: Deriving the implicit feature of App_t for training.

```
  Input: $App_t$: a training App
  Output: $IF_t$: the implicit feature of $App_t$
1 foreach $App_i$ prior than $App_t$ do
2     $IF_t[i] \leftarrow IF_t[i] + \hat{p_{it}(\delta_{it})}$ ;
3     foreach $App_m$ between $App_i$ and $App_t$ do
4         $IF_t[i] \leftarrow IF_t[i] + \hat{p_{im}(\delta_{jm})} \times IF_m[t]$ ;
5     end
6 end
7 return $IF_t$
```

For example, suppose we have an AUG as shown in Figure 2 and a usage trace as $\cdots \rightarrow App_1 \xrightarrow{1} App_2 \xrightarrow{0.5} App_1 \xrightarrow{0.5} App_3 \rightarrow \ldots$. Figure 3 shows the process of obtaining the implicit feature of App_3. We first estimate $p'^{(0)}_{13}$ from $App_1 \xrightarrow{0.5} App_3$, then $p'^{(1)}_{23}$ from $App_2 \xrightarrow{0.5} App_1 \xrightarrow{0.5} App_3$, and finally update $p'^{(2)}_{13}$ from $App1 \xrightarrow{1} App_2 \xrightarrow{0.5} App_1 \xrightarrow{0.5} App_3$. Note that $p'^{(0)}_{13}$ is reused for calculating $p'^{(1)}_{23}$, and $p'^{(1)}_{23}$ and $p'^{(0)}_{13}$ are reused for updating $p'^{(2)}_{13}$. The implicit feature of App_3 is $< 0.01, 0.13, 0 >$.

3) Implicit Features for Testing: Since the App to be predicted for current invocation, App_t, is unknown for testing, the derivation process of implicit features for training does not work. We propose an iterative refinement algorithm to estimate both App_t and its implicit feature, IF_t, for testing. Suppose θ_i is the probability of $App_t = App_i$, the implicit feature IF_t is calculated as in Equation 4 which is a linear combination of the IF of each App_i. In addition, $M = [IF_1^T, IF_2^T, \ldots]$ represents the transition matrix among Apps, where IF_1^T, IF_2^T, $\ldots$ are column vectors. Then, the value of θ_i could be updated by Equation 5, which is the probability of staying in App_i after one-step walking along the transition matrix M. We keep updating θ_i and IF_t iteratively, until App_t is fixed to one specific App. In our experiments, the iterative refinement process converges in about 3 iterations. Algorithm 2 depicts the derivation of the implicit feature for testing.

$$IF_t = \sum_{App_i} \theta_i \times IF_i \tag{4}$$

$$\theta_i = \sum_{App_m} IF_t[m] \times M[m][i] \tag{5}$$

Algorithm 2: Deriving the implicit feature for testing.

```
Input: $t$: a testing case
Output: $IF_t$: the implicit feature at $t$
1  while $iter < threshold$ do
2      foreach $\theta_j$ do
3          $IF_t \leftarrow IF_t + \theta_i \times IF_i$ ;
4      end
5      foreach $App_i$ prior than time $t$ do
6          $\theta_i \leftarrow \theta_i + IF_t[m] \times M[m][i]$ ;
7          Normalize $\theta_i$ ;
8      end
9      $iter \leftarrow iter + 1$ ;
10 end
11 return $IF_t$
```

For example, suppose the testing case is $\cdots \rightarrow App_1 \xrightarrow{1} App_2 \xrightarrow{0.5} App_1 \xrightarrow{0.5} App_t$. First, we initialize θ_i as $< 1/3, 1/3, 1/3 >$, which gives equal probability to each App, and the transition matrix $M = \begin{bmatrix} 0.49 & 0.6 & 0.01 \\ 0 & 0 & 0.13 \\ 0 & 0 & 0 \end{bmatrix}$, which is derived by calculating the IF of each App shown in Equation 3. Note that the last row is all zero because there is no App_3 transiting to any other Apps. Then, the implicit feature is $< 0.37, 0.04, 0 >$ in the first iteration. Next, θ_i is updated to $< 0.18, 0.22, 0.01 >$, and normalized as $< 0.44, 0.54, 0.02 >$ according to one-step walk in M with the calculated implicit feature as the prior probability. Then, we can obtain the implicit feature as $< 0.53, 0.01, 0 >$ in the second iteration.

IV. Personalized Feature Selection

The goal of the personalized feature selection is to use as fewer features as possible to guarantee an acceptable accuracy. Due to the energy and storage consumption of collecting sensors readings and Apps transition relations, we should select useful features for different users in advance. Furthermore, through the personalized feature selection, we could avoid the curse of dimensionality on performing the kNN. We first apply the personalized feature selection on the training data, and then only the selected features are required to be collected in the future.

Here, we propose a greedy algorithm to select the best feature iteratively. We adopt the concept of Minimum Description Length (MDL) [13] to evaluate the goodness of the features. For different features, we can have varied projections of the training data. We claim that if a feature needs fewer bits to describe its data distribution, it is good for predicting the data. Therefore, in each iteration, the feature with the minimum description length is selected. Then, those data points which are correctly predicted are logically eliminated from the training data, and the next feature is selected by the same process repeatedly. We define the description length of the hypothesis, which is shown in Equation 6, as the length of representing the training data. $NG(App_i)$ is the number of groups of App_i. The description length of Data given the hypothesis is the total number of miss-classified data which is formulated as in Equation 7.

$$L(H) = \sum_i \log_2 NG(App_i) \tag{6}$$

$$L(D|H) = \sum_i \log_2(missClassified(App_i) + 1) \tag{7}$$

For example, given 8 data points in the training data and three features as shown in Figure 4. In the first round, Time is the feature with minimum description length. Those data points marked as red are correctly predicted and will be removed. Therefore, in the second round, only two data points are left, and the feature of Wi-Fi signal is selected due to its minimum description length.

The selection process stops when a percentage of ρ of the training data is covered. Note that the number of features affects the energy and storage consumption and is set according to the capability of the smartphones. Algorithm 3 depicts the process of personalized feature selection. After the selection, only the readings of the sensors which are selected will be collected as the explicit feature in the future. In addition, only the selected Apps will be used to construct AUG.

V. Experimental study

In this section, we compare the performance of the proposed KAP framework with other existing methods including 1) most frequently used (MFU) method, 2) most recently used (MRU) method which is the built-in prediction method in most mobile OS, such as Android and iOS, 3) SVM, 4) App Naive Bayes [6], 5) Decision Tree, and 6) AdaBoost. In the following, we first discuss the collected dataset, then introduce the metrics employed to evaluate the performance, and finally deliver the experimental results. In addition, we conduct the experiments on a real world

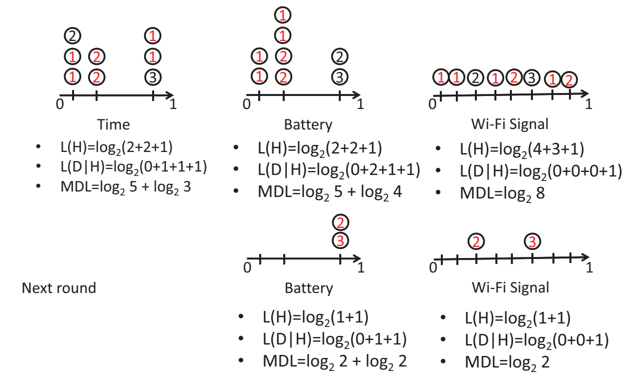


Figure 4. An example of feature selection where the red data points are correctly predicted.

Algorithm 3: Personalized feature selection.

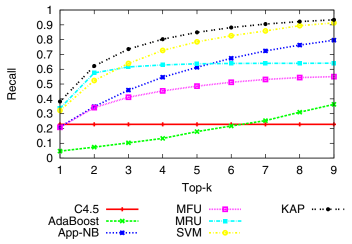
```
Input: $D_z$: the training data
Output: $PF$: the personalized features
1  Let $N_z \leftarrow |D_z|$ ;
2  while $|D_z| < \rho N_z$ do
3      foreach feature $f$ do
4          Calculate $DL_f$: description length for feature $f$ ;
5      end
6      $PF \leftarrow PF \cup \{\operatorname{argmax}_f DL_f\}$ ;
7      Let $D_a$ be the set of accurately predicted data points ;
8      $D_z \leftarrow D_z - D_a$ ;
9  end
10 return $PF$
```

dataset which contains totally 50 volunteers from June 2010 to January 2011.

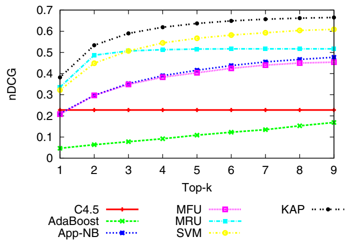
A. Experimental Results

To evaluate the performance of predicting Apps usage by the proposed KAP framework, we first evaluate the overall performance using average recall and nDCG [14] when predicting different numbers of Apps, also assess the efficiency of the proposed framework. Note that we use top-$k = 4$, kNN=40%, and the minimum data coverage of personalized feature selection ρ as 70% to be the default parameter settings throughout the experiment.

1) Overall Performance: First, we evaluate the performance KAP and other different methods under various numbers of prediction, k. As can be seen in Figure 5, when the number of prediction k increases, both the recall and nDCG values also increase. However, KAP, SVM perform better than others. In Figure 5(a), when $k = 9$ (the number of predictions shown in the latest Android system), the recall of KAP and SVM could be more than 90%, while it is below



(a) Recall



(b) nDCG

Figure 5. Impact of the number of prediction, k.

Table I
THE STORAGE CONSUMPTION AND ACCURACY UNDER VARIED DATA COVERAGE ρ.

Coverage(%)	20	40	60	80	100
Storage(KB)	28	34	43	52	94
Recall	0.78	0.80	0.82	0.82	0.83
nDCG	0.50	0.52	0.55	0.57	0.58

80% for the others. On the other hand, the nDCG value of KAP shown in Figure 5(b) is always higher than that of the other methods, which means the prediction order of KAP is better.

2) Impact of Personalized Feature Selection: For the proposed KAP method, we evaluate the performance of the personalized feature selection to see if the proposed MDL-based selection algorithm could reduce the used storage when maintaining a good prediction accuracy. For one user, the average used storage and prediction accuracy is shown in Table I under different data coverage ρ. As can be seen in Table I the personalized feature selection could reduce 55% of training data size and only lose 1% of recall and 3% of nDCG when the data coverage is 70%. In addition, Table II compares the execution time of KAP with and without the personalized feature selection, where the training time is reduced dramatically under $\rho = 70\%$.

Table II
THE EXECUTION TIME OF KAP WITH AND WITHOUT PERSONALIZED FEATURE SELECTION.

Execution time (ms)	Training	Testing	Total
KAP	86	160	246
KAP without selection	185	160	345

VI. Conclusion

In this paper, we propose an Apps usage prediction framework, KAP, which predicts Apps usage regarding both the explicit readings of mobile sensors and the implicit transition relation among Apps. For the explicit feature, we consider three different types of mobile sensors: 1) device sensors, 2) environmental sensors, and 3) personal sensors. For the implicit features, we construct an Apps Usage Graph (AUG) to model the transition probability among Apps. Then, for each training datum, we could represent the next used App as the implicit feature which describes the probability of transition from other Apps. Note that, since the next App in the testing data is unknown, we propose an iterative refinement algorithm to estimate both the probability of the App to be invoked next and its implicit feature. We claim that different usage behaviors are correlated to different types of features. Therefore, a personalized feature selection algorithm is proposed, where for each user, only the most relative features are selected. Through the feature selection, we can reduce the dimensionality of the feature space and the energy/storage consumption, and demonstrate the efficiency of MDL-based selection algorithm. We integrate the explicit and implicit features as the feature space and the next used App as the class label to perform kNN classification. In the experimental results, our method outperforms the state-or-the-art methods and the currently used methods in most mobile devices.

VII. Acknowledgement

Wen-Chih Peng was supported in part by HTC, by the National Science Council, Project No.102-2221-E-009-171-MY3 and 100-2218-E-009-016-MY3, by Academic Sinica Theme project, Project No. AS-102-TPA06, by Taiwan MoE ATU Program, and by Microsoft.

References

[1] P. Yin, P. Luo, W.-C. Lee, and M. Wang, "App recommendation: a contest between satisfaction and temptation," in *Sixth ACM International Conference on Web Search and Data Mining, WSDM 2013, Rome, Italy, February 4-8, 2013*, 2013, pp. 395–404.

[2] T. M. T. Do, J. Blom, and D. Gatica-Perez, "Smartphone usage in the wild: a large-scale analysis of applications and context," in *Proceedings of the 13th International Conference on Multimodal Interfaces, ICMI 2011, Alicante, Spain, November 14-18, 2011*, 2011, pp. 353–360.

[3] K. Shi and K. Ali, "Getjar mobile application recommendations with very sparse datasets," in *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*, 2012, pp. 204–212.

[4] B. Yan and G. Chen, "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 01, 2011*, 2011, pp. 113–126.

[5] Z.-X. Liao, P.-R. Lei, T.-J. Shen, S.-C. Li, and W.-C. Peng, "Mining temporal profiles of mobile applications for usage prediction," in *12th IEEE International Conference on Data Mining Workshops, ICDM Workshops, Brussels, Belgium, December 10, 2012*, 2012, pp. 890–893.

[6] C. Shin, J.-H. Hong, and A. K. Dey, "Understanding and prediction of mobile application usage for smart phones," in *The 2012 ACM Conference on Ubiquitous Computing, Ubicomp '12, Pittsburgh, PA, USA, September 5-8, 2012*, 2012, pp. 173–182.

[7] T. Yan, D. Chu, D. Ganesan, A. Kansal, and J. Liu, "Fast app launching for mobile devices using predictive user context," in *The 10th International Conference on Mobile Systems, Applications, and Services, MobiSys'12, Ambleside, United Kingdom - June 25 - 29, 2012*, 2012, pp. 113–126.

[8] S. Chen, J. L. Moore, D. Turnbull, and T. Joachims, "Playlist prediction via metric embedding," in *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*, 2012, pp. 714–722.

[9] Z.-X. Liao, W.-C. Peng, and P. S. Yu, "Mining usage traces of mobile applications for dynamic preference prediction," in *17th Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD 2013, Gold Coast, Australia, April 13-17, 2013*, 2013.

[10] P.-R. Lei, T.-J. Shen, W.-C. Peng, and I.-J. Su, "Exploring spatial-temporal trajectory model for location prediction," in *12th IEEE International Conference on Mobile Data Management, MDM 2011, Luleå, Sweden, June 6-9, 2011, Volume 1*, 2011, pp. 58–67.

[11] Y. Dong, J. Tang, S. Wu, J. Tian, N. V. Chawla, J. Rao, and H. Cao, "Link prediction and recommendation across heterogeneous social networks," in *12th IEEE International Conference on Data Mining, ICDM 2012, Brussels, Belgium, December 10-13, 2012*, 2012, pp. 181–190.

[12] J. Kogan, "Feature selection over distributed data streams through convex optimization," in *Proceedings of the Twelfth SIAM International Conference on Data Mining, Anaheim, California, USA, April 26-28, 2012*, 2012, pp. 475–484.

[13] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, pp. 465–471, 1978.

[14] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM Trans. Inf. Syst.*, vol. 20, no. 4, pp. 422–446, 2002.