# Mobile App Classification with Enriched Contextual Information

Hengshu Zhu, Huanhuan Cao, Enhong Chen, *Senior Member, IEEE,* Hui Xiong, *Senior Member, IEEE,* and Jilei Tian

**Abstract**—The study of the use of mobile Apps plays an important role in understanding the user preferences, and thus provides the opportunities for intelligent personalized context-based services. A key step for the mobile App usage analysis is to classify Apps into some predefined categories. However, it is a nontrivial task to effectively classify mobile Apps due to the limited contextual information available for the analysis. For instance, there is limited contextual information about mobile Apps in their names. However, this contextual information is usually incomplete and ambiguous. To this end, in this paper, we propose an approach for first enriching the contextual information of mobile Apps by exploiting the additional Web knowledge from the Web search engine. Then, inspired by the observation that different types of mobile Apps may be relevant to different *real-world contexts*, we also extract some contextual features for mobile Apps from the context-rich device logs of mobile users. Finally, we combine all the enriched contextual information into the Maximum Entropy model for training a mobile App classifier. To validate the proposed method, we conduct extensive experiments on 443 mobile users' device logs to show both the effectiveness and efficiency of the proposed approach. The experimental results clearly show that our approach outperforms two state-of-the-art benchmark methods with a significant margin.

**Index Terms**—Mobile App classification, Web knowledge, Real-world contexts, Enriched contextual information.

## 1 INTRODUCTION

With the wide spread use of mobile devices in recent years, a huge number of mobile Apps have been developed for mobile users. For example, as of the end of July 2013, there are more than 1.9 million Apps and 100 billion cumulative downloads at Apple's App store and Google Play. Indeed, mobile Apps play an important role in the daily lives of mobile users. Intuitively, the study of the use of mobile Apps can help to understand the user preferences, and thus motivates many intelligent personalized services, such as App recommendation, user segmentation and target advertising [17], [19], [20], [29], [34].

However, the information directly from mobile Apps is usually very limited and ambiguous. For example, a user's preference model may not fully understand the information "*the user usually plays Angry Birds*" unless the mobile App "*Angry Birds*" is recognized as a predefined App category "*Game/Stategy Game*". Indeed, due to the large number and high increasing speed of mobile Apps, it is expected to have an effective and automatic approach for mobile App classification. Nonetheless, one may argue that some mobile Apps are associated with predefined tags or descriptions as metadata in the App delivery platform (e.g., App Stores) and these data can be directly used for recognizing the latent semantic meanings. However, these data may be difficult to obtain by the third party services, especially in the case that there exist multiple App delivering channels and it is not able to track the source of a mobile App, such as the practical scenario of the Android ecosystem. Also, those tags are usually not very accurate to reflect the latent semantic meanings behind the use of mobile Apps. For example, a security mobile App "*Safe 360*" is tagged as "*Business*" in the Nokia Store [3], which is obviously too general to capture the latent semantic meaning for understanding the real App usage.

Indeed, mobile App classification is not a trivial task which is still under-development. The major challenge is that there are not many effective and explicit features available for classification models due to the limited contextual information of Apps available for the analysis. Specifically, there is limited contextual information about mobile Apps in their names, and the only available explicit features of mobile Apps are the semantic words contained in their names. However, these words are usually too short and sparse to reflect the relevance between mobile Apps and particular categories. For example, Figure 1 shows the distribution of the number of mobile Apps with respect to the name length in our real-world data set. In this figure, we can observe that the distribution roughly follows the power law, and most Apps only contain less than three words in their names.

- *H. Zhu and E. Chen are with the School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui 230026, China.*
  *Email: zhs@mail.ustc.edu.cn; cheneh@ustc.edu.cn*
- *H. Cao and J. Tian are with the Nokia Research Center, Beijing, 100010, China.*
  *Email: happia.cao@gmail.com; jilei.tian@nokia.com*
- *H. Xiong is with the Management Science and Information Systems Department, Rutgers Business School, Rutgers University, Newark, NJ 07102 USA.*
  *Email: hxiong@rutgers.edu*

*This is a substantially extended and revised version of [32], which appears in Proceedings of the 21st ACM Conference on Information and Knowledge Management (CIKM2012).*

Digital Object Indentifier 10.1109/TMC.2013.113 1536-1233/13/$31.00 © 2013 IEEE

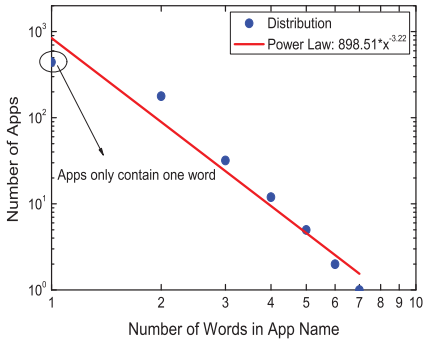



Fig. 1. The distribution of the number of mobile Apps with respect to the name length in our real-world date set.

To this end, in this paper, we propose to leverage both Web knowledge and *real-world contexts* for enriching the contextual information of Apps, thus can improve the performance of mobile App classification. According to some state-of-the-art works on short text classification [9], [11], [23], [25], [27], an effective approach for enriching the original few and sparse textual features is leveraging Web knowledge. Inspired with those works, we propose to take advantage of a Web search engine to obtain some snippets to describe a given mobile App for enriching the textual features of the App. The leveraged Web search engine can be a general search engine such as Google or the vertical App search engine provided by an App store. However, sometimes it may be difficult to obtain sufficient Web knowledge for new or rarely used mobile Apps. In this case, the relevant real-world contexts of mobile Apps may be useful. Some observations from the recent studies [14], [17], [19], [28], [29], [31] indicate that the App usage of a mobile user is usually context-aware. For example, business Apps are likely used under the context like "*Location: Work Place*", "*Profile: Meeting*", while games are usually played under the context like "*Location: Home*", "*Is a holiday?: Yes*". Compared with Web knowledge, the relevant real-world contexts of new or rarely used mobile Apps may be more available since they can be obtained from the context-rich device logs of the users who used them in mobile devices. Therefore, we also propose to leverage the relevant real-world contexts of mobile Apps to improve the performance of App classification. To be specific, the contributions of this paper are summarized as follows.

First, automatic mobile App classification is a novel problem which is still under-development. To the best of our knowledge, we are one of the first attempts to study this problem. Furthermore, we are the first to leverage both Web knowledge and relevant real-world contexts to enrich the limited contextual information of mobile Apps for solving this problem.

Second, we study and extract several effective features from both Web knowledge and real-world contexts through the state-of-the-art data mining technologies. Then, we propose to exploit the Maximum Entropy model (MaxEnt) [7], [22] for combining the effective features to train a very effective and efficient App classifier.

Finally, to evaluate the proposed approach, we conduct extensive experiments on the context-rich mobile device logs collected from 443 mobile users, which contain 680 unique mobile Apps and more than 8.8 million App usage records. The experimental results clearly show that our approach outperforms two state-of-the-art benchmark approaches with a significant margin.

**Overview.** The remainder of this paper is organized as follows. In Section 2, we provide a brief review of related works. Section 3 presents an overview of the proposed approach and some preliminaries. In Section 4, we give the technical details of extracting Web knowledge based features and real-world contextual features, respectively. Furthermore, we also introduce the machine learning model for training App classifier. Section 5 shows the experimental results based on a real-world data set. Finally, we conclude this paper in Section 6.

## 2 Related Work

Automatic mobile App classification is a novel application problem, however, it also can be regarded as the problem of classifying short & sparse texts. Short & sparse texts are very common in real-world services, such as query terms and SMS, which often contain limited and sparse textual information for utilizing. Compared with traditional text classification tasks, classifying short & sparse text is very challenging and thus attracts many researchers' attention. For example, Phan *et al.* [23] proposed to leverage hidden topics to improve the representation of short & sparse text for classification. The hidden topics are learnt from external data set with seeds selection to avoid noise, such as Web knowledge. Sahami *et al.* [25] proposed a novel similarity measuring approach for short text snippets, which can also be proven by a kernel function. Specifically, this approach utilizes a Web search engine to enrich original textual information, which can be leveraged for short & sparse text classification. Furthermore, Yih *et al.* [30] improved the measuring approach by exploiting an additional learning process to make the measurement more efficient. Broder *et al.* [9] proposed to extract information from the top related search results of the query from a Web search engine, and Shen *et al.* [27] studied using a Web directory to classify queries. Cao *et al.* [11] proposed to use Web knowledge for enriching both the contextual features and local features of Web queries for query classification.

Indeed, some of above techniques can be leveraged for our App classification task. For example, recently, according to Cao's work [11], Ma *et al.* [20] proposed an automatic approach for normalizing user App usage records, which can leverage search snippets to build vector space for both App usages and categories, and classify App usage records according to the Cosine space distance. Compared with these works, the work reported in this paper does not only comprehensively take advantage of more Web knowledge based features but also leverages the relevant contexts of mobile Apps which reflect their usage patterns from user perspective.

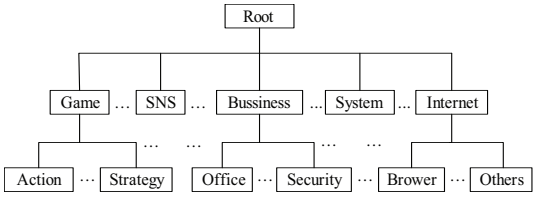



Fig. 2. An example of the mobile App taxonomy.

In recent years, with rapid development of mobile devices, many researchers studied leveraging real-world contexts to improve traditional services, such as personalized context-aware recommendation [17], [19], [29], [34], context-aware user segmentation [20] and user context-aware tour guide [14], [28]. As a result, researchers have found many user behaviors are usually context-aware, that is, some user behaviors are more likely to appear under a particular context. For example, Cao *et al.* [10] proposed an efficient approach for mining associations between user App usage and real-world contexts. A different metric to count support is developed for addressing the unbalanced occurrences of App usage records and context data. Bao *et al.* [6] studied leveraging unsupervised approaches for modeling user context and App usage. In this work, the raw context data and App usage records are first segmented and then modeled by topic models. Ma *et al.* [20] studied how to leverage the associations between contexts and user activities for discovering similar users with respect to habit by addressing the sparseness of such associations. Inspired by these works, we argue that the types of mobile Apps that a user will use may be relevant to his (or her) contexts. Thus, in this paper we propose to leverage the relevant contextual information of mobile Apps for improving the performance of App classification.

## 3 Overview

Here, we introduce several related notions and give an overview of our mobile App classification approach.

### 3.1 Preliminary

• **App Taxonomy.** To recognize the semantic meanings of Apps, we can classify each App into one or more categories according a predefined *App taxonomy*. Specifically, an App taxonomy $\Upsilon$ is a tree of categories where each node corresponds to a predefined App category. The semantic meaning of each App can be defined by the category labels along the path from the root to the corresponding nodes. Figure 2 shows a part of the App taxonomy used in our experiments.

• **Search Snippets.** In our approach, we propose to leverage the Web knowledge to enrich the textual information of Apps. To be specific, we first submit each App name to a Web search engine (e.g., Google or other App search engines), and then obtain the *search snippets* as the additional textual information of the corresponding App. A search snippet is the abstract of the Web page which are returned as relevant to the submitted search query. The textual information in search snippets is brief but

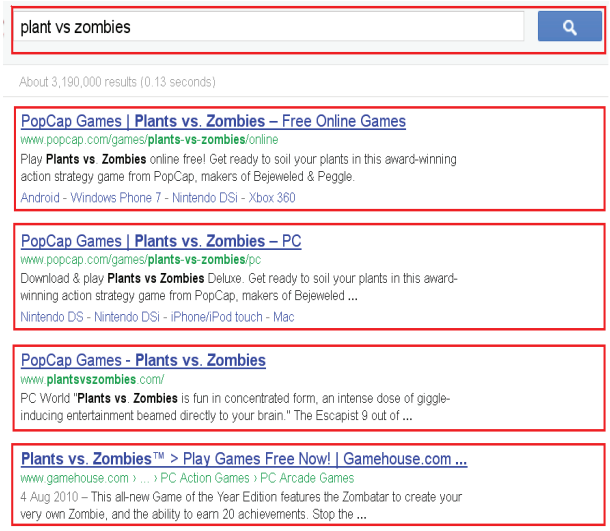



Fig. 3. The snippets in the result pages from Google.

can effectively summarize the corresponding web pages. Thus, they are widely used for enriching the original textual information in the short text classification problem. Figure 3 shows some examples of search snippets for the App "*Plant Vs. Zombies*" from Google.

• **Context Log.** Smart mobile devices can capture the historical context data and the corresponding App usage records of users through context-rich device logs, or *context logs* for short. For example, Table 1 shows an example of context log which contains several *context records*, and each context record consists of a timestamp, the most detailed contextual information at that time, and the corresponding App usage record captured by the mobile device. The contextual information at a time point is represented by several *contextual features* (e.g., *Day name*, *Time range*, and *Location*) and their corresponding values (e.g., *Saturday*, *AM8:00-9:00*, and *Home*), which can be annotated as *contextual feature-value pairs*. Moreover, App usage records can be empty (denoted as "Null") because users do not always use Apps. In Table 1, location related raw data in the context logs, such as GPS coordinates or cell IDs, have been transformed into semantic locations such as "*Home*" and "*Work Place*" by a location mining approach [20]. The basic idea of such approach is to find the clusters of user positions and recognize their semantic meanings through the time pattern analysis.

### 3.2 Overview of Our Approach

The proposed approach for mobile App classification consists of two main stages. First, we collect many context logs from mobile users, and extract both Web knowledge based features and real-world contextual features for the Apps appearing in these logs. Second, we take advantage of the machine learning model for training an App classifier. Figure 4 illustrates the main framework of the proposed approach. To be specific, given a target taxonomy $\Upsilon$, an App $a$ and a system-specified parameter $K$, our approach incorporates the features extracted from both the relevant Web knowl-

TABLE 1
An example of context log from a mobile user in our real-world data set.

| Timestamp | Context | App usage records |
|---|---|---|
| $t_1$ | {(Day name: Monday),(Time range: AM8:00-9:00),(Profile: General),(Battery Level: High),(Location: Home)} | Angry Birds |
| $t_2$ | {(Day name: Monday),(Time range: AM8:00-9:00),(Profile: General),(Battery Level: High),(Location: On the Way)} | Null |
| $t_3$ | {(Day name: Monday),(Time range: AM8:00-9:00),(Profile: General),(Battery Level: High),(Location: On the Way)} | Twitter |
| ..... | | |
| $t_{55}$ | {(Day name: Monday),(Time range: AM8:00-9:00),(Profile: General),(Battery Level: High),(Location: On the Way)} | UC Web |
| $t_{56}$ | {(Day name: Monday),(Time range: AM8:00-9:00),(Profile: General),(Battery Level: High),(Location: On the Way)} | Null |
| $t_{57}$ | {(Day name: Monday),(Time range: AM8:00-9:00),(Profile: General),(Battery Level: High),(Location: On the Way)} | Music Player |
| ..... | | |
| $t_{359}$ | {(Day name: Monday),(Time range: AM10:00-11:00),(Profile: Meeting),(Battery Level: High),(Location: Work Place)} | Null |
| $t_{360}$ | {(Day name: Monday),(Time range: AM10:00-11:00),(Profile: Meeting),(Battery Level: High),(Location: Work Place)} | SMS |

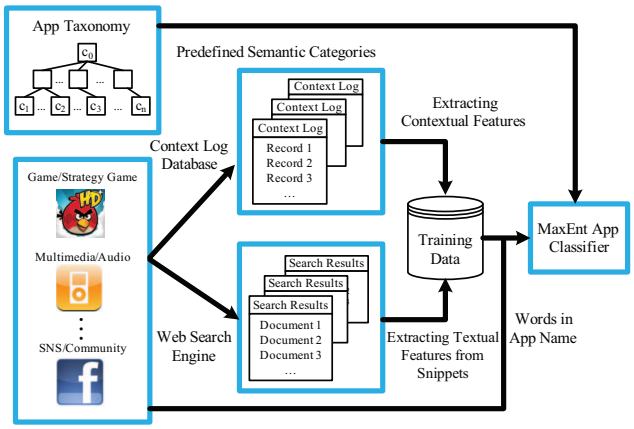



Fig. 4. The framework of our App classification approach.

edge and contextual information of $a$ to classify $a$ into a ranked list of $K$ categories $c_1^a, c_2^a, ..., c_K^a$, among $N_c$ leaf categories $\{c_1, ..., c_{N_c}\}$ of $\Upsilon$.

When we utilize the machine learning model to train App classifiers, the most important work is to select effective features. Intuitively, given an App $a$ and its category label $c$, the basic features that can reflect the relevance between $a$ and $c$ are the words contained in the name of $a$. Suppose that the name of App $a$ consists of a set of words $\{w_i^a\}$, each $w_i^a$ can be considered as a relevant boolean feature to the category label $c$. That is, the occurrence of a word $w$ can be denoted as $f(w) = 1$ while vise versa we denote $f(w) = 0$. The weights of these features can be learned in the training process of the machine learning model.

A critical problem of this type of features is that the lengths of App names are usually too short and the contained words are very sparse. As a result, it is difficult to train an effective classifier by only taking advantage of the words in App names. Moreover, the available training data are usually with limited size and may not cover a sufficiently large set of words for reflecting the relevance between Apps and category labels. Therefore, a new App whose partial, or all words in name do not appear in the training data will not obtain accurate classification results if the classifier is only based on the words in App names. To this end, we should also take consideration of other effective features which can capture the relevance between Apps and category labels. In the following section, we introduce the features extracted from both the relevant Web knowledge and real-world contextual information for training an App classifier.

## 4 Extracting Effective Features for App Classification

In this section we first introduce how to extract effective textual and contextual features for App classification from both Web knowledge and real-world contextual information, respectively. Then we also introduce how to integrate these features into a state-of-the-art machine learning model for training App classifier.

### 4.1 Extracting Web based Textual Features

In this subsection, we introduce how to leverage Web knowledge for extracting additional textual features of mobile Apps. To be specific, we investigate two such kinds of textual features to capture the relevance between Apps and the corresponding category labels, namely, *Explicit Feedback of Vector Space Model* and *Implicit Feedback of Semantic Topics*.

#### 4.1.1 Explicit Feedback of Vector Space Model

This type of features exploits the top $M$ results (i.e., search snippets) returned by a Web search engine through leveraging the explicit feedback of Vector Space Model (VSM) [26]. Given an App $a$ and its category label $c$, we first submit $a$'s name to a Web search engine (e.g., via Google API in our experiments). Then, for each of the top $M$ results, we map it to a category label in the App taxonomy $\Upsilon$ by building a Vector Space Model. There are three steps in the process of mapping snippets to categories by VSM.

First, for each App category $c$, we integrate all top $M$ snippets returned by a Web search engine for some pre-selected Apps labeled with $c$ as a *category profile* $d_c$. Particularly, we remove all stop words (e.g., "of", "the") in $d_c$ and normalize verbs and adjectives (e.g., "$plays \rightarrow play$", "$better \rightarrow good$").

Second, we build a normalized words vector $\overrightarrow{w_c} = dim[n]$ for each App category $c$, where $n$ indicates the number of all unique normalized words in all category profiles. To be specific, here we have

$$dim[i] = \frac{freq_{i,c}}{\sum_i freq_{i,c}} \quad (1 \leq i \leq n), \tag{1}$$

where $freq_{i,c}$ indicates the frequency of the $i$-th word in the category profile $d_c$.

Finally, for each snippet $s$ returned for App $a$, we remove the words which do not appear in any category profile and build its normalized word vector $\overrightarrow{w_{a,s}}$ in

a similar way. Then we calculate the Cosine distance between $\overrightarrow{w_{a,s}}$ and $\overrightarrow{w_c}$ as similarity, that is,

$$Similarity(\overrightarrow{w_{a,s}}, \overrightarrow{w_c}) = \frac{\overrightarrow{w_{a,s}} \cdot \overrightarrow{w_c}}{||\overrightarrow{w_{a,s}}|| \cdot ||\overrightarrow{w_c}||}. \quad (2)$$

According to the similarity, we can map each snippet $s$ to the category $c^*$, which satisfies,

$$c^* = \arg\max_c Similarity(\overrightarrow{w_{a,s}}, \overrightarrow{w_c}). \quad (3)$$

Through the VSM, we can calculate a *general label confidence score* introduced in [11] by:

$$GConf(c, a) = \frac{M_{c,a}}{M}, \quad (4)$$

where $M_{c,a}$ indicates the number of returned related search snippets of $a$ whose category labels are $c$ after mapping. Intuitively, the $GConf$ score reflects the confidence that $a$ is labeled as $c$ gained from Web knowledge. The larger the score, the higher the confidence.

However, sometimes $GConf$ score may not accurately validate the relevance between $c$ and $a$ due to the noise category labels contained in the mapping list. In practice, we find the more unique category labels contained in the mapping list, the more uncertainty for classification. Therefore, we further define another score named *general label entropy* to measure the uncertainty of App classification, which can be calculated as follows:

$$GEnt(c, a) = -\sum_{c_i \neq c} P(c_i) \log P(c_i), \quad (5)$$

where $P(c_i) = GConf^*(c_i, a) = \frac{M^{\neg c}_{c_i,a}}{M^{\neg c}}$, where $M^{\neg c}$ is the number of returned documents without category label $c$. Intuitively, the $GEnt$ score implies the effectiveness of $GConf$ score.

#### 4.1.2 Implicit Feedback of Semantic Topics

Although the explicit feedback of VSM can capture the relevance between App and category label in terms of the occurrences of words, it does not take consideration of the latent semantic meanings behind words and may not work well in some cases. For example, in VSM, the following words "Game", "Play" and "Funny" are treated as totally different measures to calculate the distance between word vectors. However, these words indeed have latent semantic relationships because they can be categorized into the same semantic topic "Entertainment". According to [23], the latent semantic topics can improve the performance of short & sparse text classification. Thus, here we study the textual features which consider the implicit feedback of semantic topics.

To be specific, we propose to leverage the widely used Latent Dirichlet Allocation (LDA) model [8] for learning latent semantic topics. Therefore, according to LDA, a category profile $d_c$ is assumed to be generated as follows. First, before generating a category profile, $K$ prior conditional distributions of words given latent topics $\{\phi_z\}$ are generated from a prior Dirichlet distribution $\beta$. Second, a prior latent topic distribution $\theta_c$ is generated from a prior Dirichlet distribution $\alpha$ for each category $c$. Then, for generating the $j$-th word in $d_c$ denoted as $w_{c,j}$, the model firstly generates a latent topic $z$ from $\theta_c$ and then generates $w_{c,j}$ from $\phi_z$. Figure 5 shows the graphical model of LDA.

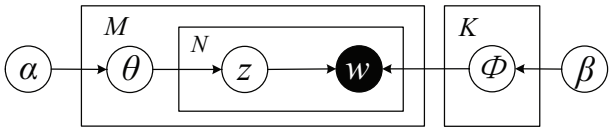



Fig. 5. The graphical model of LDA, where M, N, and K are the number of category profiles, words, and latent topics, respectively; $\alpha$ and $\beta$ are the prior parameters.

The process of training LDA model is to learn proper latent variables $\theta$ and $\phi$ for maximizing the posterior distribution of category profiles, i.e., $P(d_c|\alpha, \beta, \theta, \phi)$. In this paper, we take advantage of a Markov chain Monte Carlo method named Gibbs sampling [15] for training LDA model. This method begins with a random assignment of latent topics to words for initializing the state of Markov chain. In each of the following iterations, the method will re-estimate the conditional probability of assigning a latent topic to each word, which is conditional on the assignment of all other words. Then a new assignment of latent topics to words according to those latest calculated conditional probabilities will be scored as a new state of Markov chain. Finally, after several rounds of iterations, the assignment will converge, which means each word is assigned a final latent topic.

After learning latent topics, we extract the features based on the implicit feedback of semantic topics as follows. Given an App $a$, we first leverage a Web search engine to obtain the top $M$ relevant snippets and remove the words which do not appear in any category profile. Then, we map each snippet $s$ to a category by calculating the KL-divergence between their topic distributions as:

$$D_{KL}(P(z|s)||P(z|c)) = \sum_k P(z_k|s) ln \frac{P(z_k|s)}{P(z_k|c)}, \quad (6)$$

where $P(z_k|c) = P(z_k|d_c)$ and $P(z_k|s) \propto P(z) \prod P(w_s|z)$ can be obtained through the LDA training process. The category $c^*$ with the smallest KL-divergence will be selected as the label of $s$, that is,

$$c^* = \arg\min_c D_{KL}(P(z|s)||P(z|c)). \quad (7)$$

Finally, we calculate the *topic confidence score* for a given category $c$ as follows:

$$TConf(a, c) = \frac{T_{a,c}}{M}, \quad (8)$$

where $T_{a,c}$ indicates the number of returned snippets for $a$ with the category label $c$. Intuitively, the $TConf$ score reflects the confidence that $a$ is labeled as $c$ with respect to latent semantic topics. The larger the score, the higher the confidence. Moreover, we also calculate the *topic based general label entropy* in similar way according to Equation 5.

### 4.2 Extracting Real-World Contextual Features

In this subsection, we introduce how to extract effective contextual features of mobile Apps from real-world context logs. To be specific, we study three types of contextual features, namely, *Pseudo Feedback of Context Vectors*, *Implicit Feedback of Context Topics* and *Frequent Context Patterns*.

#### 4.2.1 Pseudo Feedback of Context Vectors

The first type of contextual features considers the feedback of context vectors. We assume that the usage of a particular category of Apps is relevant to some contextual feature-value pairs. To be specific, given the Apps in the "*Game/Strategy Game*" category, they may be relevant to the contextual feature-pairs (*Day period: Evening*) and (*Location: Home*), respectively. Based on this assumption, similar to the VSM-based approach introduced in Section 4.1.1, we build a context vector for each App category as follows.

First, for each pre-selected and labeled App $a$, we collect all context records which record the usage of App $a$ from the context logs of many mobile users.

Second, we extract the contexts in these context records which consist of contextual feature-value pairs and build a *context profile* $R_a = \{(p_i, freq_{i,a})\}$ for each App $a$ from these contexts, where $p_i$ denotes a contextual feature-value pair appearing in these contexts and $freq_{i,a}$ indicates the corresponding frequency. Similarly, we can build the context profile $R_c$ for each category $c$ by combining all the context profiles of the pre-selected Apps labeled with $c$.

Last, we can define the context vector of App $a$ as $\overrightarrow{\Omega_a} = dim[m]$, where $m$ indicates the total number of unique contextual feature-value pairs and $dim[i] = \frac{freq_{i,a}}{\sum_i freq_{i,a}} (1 \leq i \leq m)$. Similarly, we can build a context vector $\overrightarrow{\Omega_c}$ for each category $c$ according to $R_c$.

After building the context vectors for App categories, we can take the feedback of the pseudo category based on context similarity as a contextual feature. To be specific, given an App $a$ and a category label $c$, we first build the context vector of $a$ denoted as $\overrightarrow{\Omega_a}$ and then calculate the Cosine distance between $\overrightarrow{\Omega_a}$ and all App categories' context vectors. Finally, we rank category labels in descending order according to their Cosine similarity to $\overrightarrow{\Omega_a}$. Particularly, we define the pseudo category $c^*$ by $c^* = \arg\max_c Similarity(\overrightarrow{\Omega_a}, \overrightarrow{\Omega_c})$ and calculate the *category rank distance* by:

$$CRDistance(a, c) = Rk(c) - Rk(c^*) = Rk(c) - 1, \quad (9)$$

where $Rk(c)$ denotes the rank of category $c$ obtained by comparing vector distances to $a$. Intuitively, the $CRDistance \in [0, N_c)$, where $N_c$ indicates the number of leaf nodes in the App taxonomy $\Upsilon$. Obviously, the smaller the distance, the more likely the category label $c$ is the correct label.

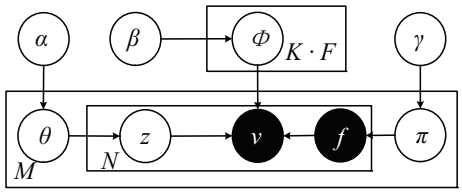



Fig. 6. The graphical model of LDAC, where M, N, F, K are the number of context profiles, contextual feature-value pairs, contextual features, and latent context topics, respectively; $\alpha$, $\beta$ and $\gamma$ are the prior parameters.

#### 4.2.2 Implicit Feedback of Context Topics

Although the pseudo feedback of context vectors can capture the relevance between Apps and category labels in terms of the occurrences of contextual feature-value pairs, it does not take consideration of the latent semantic meanings behind contextual information. Similarly as discussed in Section 4.1.2, we observe that many contextual feature-value pairs have some latent semantic meanings, e.g., (*Day period: Evening*), (*Is a holiday?: Yes*) and (*Location: Home*) may all imply the latent *context topic* "Relax". Intuitively, these context topics may capture the relationships between Apps and category labels more accurately. For example, we observe that "Games" are often played in the contexts belong to topic "Relax", while "Business Apps" are often used in the contexts belong to topic "Working". Thus, here we also investigate the implicit feedback of context topics for App classification.

An intuitive approach for discovering these context topics is leveraging topic models, i.e., take the context profile $R_c$ of each category $c$ as document and each contextual feature-value pairs as words. However, a critical challenge when utilizing existing topic models, e.g., LDA, for modeling contexts is that context values are not only influenced by latent context topics but also by context features. For example, BlueTooth information can only be got when user opens BlueTooth sensor, and location information often cannot be obtained in underground subways due to the lack of GPS/cell ID information. Therefore, to accurately model context information, in this paper we also leverage the extended Latent Dirichlet Allocation on Context model (LDAC) [6] for mining latent context topics.

In the LDAC model, a context profile $R_c$ of category $c$ is generated as follows. Firstly, a prior context topic distribution $\theta_{R_c}$ is generated from a prior Dirichlet distribution $\alpha$. Secondly, a prior contextual feature distribution $\pi_{R_c}$ is generated from a prior Dirichlet distribution $\gamma$. Then, for the $i$-th contextual feature-value pair in $R_c$, a context topic $z_{R_c,i}$ is generated from $\theta_{R_c}$, a contextual feature $f_{R_c,i}$ is generated from $\pi_{R_c}$, and the value of $f_{R_c,i}$ denoted as $v_{R_c,i}$ is generated from the distribution $\phi_{z_{R_c,i}, f_{R_c,i}}$. Moreover, there are totally $K \times F$ prior distributions of contextual feature-value pairs $\{\phi_{k,f}\}$ which follow a Dirichlet distribution $\beta$. Figure 6 shows the graphical representation of the LDAC model. According to the model, we have

TABLE 2
A context topic $z$ learnt by LDAC from our real-world data set, where each $P(p|z) > 0.5$.

| |
|---|
| (Is a holiday?: Yes) |
| (Day name: Saturday) |
| (Day name: Sunday) |
| (Day period: Evening) |
| (Time range: PM20:00-21:00) |
| (Time range: PM21:00-22:00) |
| (Location: Home) |
| (Profile: General) |
| (Charging state: Charging) |
| (Charging state: Complete) |
| (Battery level: Level 6) |
| (Battery level: Level 7 (Full)) |

$$P(R_c, \theta_{R_c}, z_{R_c}, \pi_{R_c}, \Phi|\alpha, \beta, \gamma) = P(\theta_{R_c}|\alpha)P(\Phi|\beta)P(\pi_{R_c}|\gamma) \times \left(\prod_{i=1}^{N} P(v_{R_c,i}|z_{R_c,i}, f_{R_c,i}, \Phi)P(f_{R_c,i}|\pi_{R_c})P(z_{R_c,i}|\theta_{R_c})\right),$$

where $\Phi = \{\phi_{k,f}\}$ and $z_{R_c} = \{z_{R_c,i}\}$. In this paper, we leverage the Gibbs sampling based approach introduced in [6] for training LDAC model. After training process, we can obtain the probabilities $P(p|z)$ and $P(z|R_c)$, where $p$ is the contextual feature-value pair. Table 2 shows an example of a context topic learnt by LDAC from our real-world data set, which may indicate the context of relax time.

Given an App $a$, we first obtain its context profile $R_a$ from historical context log database. Then for each category label $c$, we calculate the KL-divergence between their topic distributions:

$$D_{KL}(P(z|R_a)||P(z|R_c)) = \sum_k P(z_k|R_a) ln \frac{P(z_k|R_a)}{P(z_k|R_c)}, \quad (10)$$

where $P(z|R_a) \propto P(z)\prod P(p|z)$ $(p \in R_a)$. Finally, we rank category labels in ascending order according to their KL-divergence. Particularly, we define the pseudo category $c^*$ by $c^* = \arg\min_c D_{KL}(P(z|R_a)||P(z|R_c))$, and for each given category label $c$ we calculate the *Topical Rank Distance* by:

$$TRDistance(a, c) = Rk(c) - Rk(c^*) = Rk(c) - 1, \quad (11)$$

where $Rk(c)$ denotes the rank of category label $c$ obtained by comparing KL-divergences. Intuitively, the $TRDistance \in [0, N_c)$, where $N_c$ indicates the number of category labels. Obviously, the smaller the distance, the more likely $c$ is the correct label.

#### 4.2.3 Frequent Context Patterns

When leveraging above contextual features, we regard each unique context feature-value pair as an independent measure for the relevance between contexts and the usage of a particular category of Apps. However, recently some researchers pointed out that some context feature-value pairs are mutually related rather than separate elements and their co-occurrences are relevant to App usage as well [10]. To be specific, given a context "{(*Day period: Evening*), (*Location: Home*)}" and a record of the usage of App $a$, the usage of $a$ may be relevant to the co-occurrence of feature-pairs (*Day period: Evening*) and (*Location: Home*) but not relevant to (*Time range: PM10:00-11:00*) or (*Location: Home*) separately. Along this line, we study a contextual feature to capture the relevance between the co-occurrence of contextual feature-value pairs and the App usage. Specifically, we take advantage of the *frequent context patterns* for App classification as follows.

TABLE 3
Examples of mined frequent context patterns.

| | |
|---|---|
| #1 | (*Is a holiday? Yes*)(*Day period: Evening*)(*Location: Home*) ⇒ Angry Birds (**Game/Strategy Game**) |
| #2 | (*Day period: Morning*)(*Location: Work Place*) ⇒ Yahoo Mail (**Communication/Mail&SMS**) |
| #3 | (*Day period: Evening*)(*Location: On the Way*)(*Profile: Silent*) ⇒ Music Player (**Multimedia/Audio**) |
| #4 | (*Day period: Afternoon*)(*Location: On the Way*) ⇒ Ovi Map (**Navigation/Maps**) |

According to the introduction in Section 3, given an App $a$ and many context logs $\{l\}$, we can find some combinations of contextual feature-value pairs which are relevant to the usage of $a$ as frequent context patterns from $\{l\}$. However, it is not a trivial work to mine these context patterns. As pointed out by Cao *et al.* [10], the amounts of context data and App usage records are usually extremely unbalanced, which makes it difficult to mine such context patterns through traditional association rule mining approaches. An alternative approach is only leveraging the context records with non-empty App usage records. However, it will lose the discriminative information on how likely no App will be used under a particular context. Fortunately, some researchers have studied this problem and proposed some novel algorithms for mining such context patterns. For example, Cao *et al.* [10] proposed a novel algorithm called GCPM (Generating Candidates for behavior Pattern Mining) for mining such context patterns, which are referred to as behavior patterns in their work, by utilizing different ways of calculating supports and confidences. In an incremental work of [10], Li *et al* [18] proposed a more efficient algorithm named BP-Growth for solving this problem. In this paper, we leverage the BP-Growth algorithm for mining frequent context patterns. The basic idea of the algorithm is partitioning the original context logs into smaller sub-context logs for reducing the mining space and mining frequent context patterns in these sub-context logs. Table 3 illustrates some examples of frequent context patterns mined from context logs.

It is worth noting that the mining is performed on individual users' context logs because merging all context logs may normalize the relevance between contexts and App usage. For example, given several users who usually play action games in buses and several users usually play other games in buses. If we try to discover the relevance between contexts and App usage by merging all users' context logs, we may falsely conclude that "*In a bus*" has no significant relevance with the usage of any category of Apps. In contrast, we can discover "*In a bus*" is both relevant to action games and

other games according to different people by taking into account each user's context log separately. After mining frequent context patterns, we regard the occurrences of relevant frequent context patterns as boolean features for determining a proper category label for $a$ in a similar way of leveraging the words in App names.

### 4.3 Training Mobile App Classifier

After extracting both textual and contextual features, the remaining work is to train an efficient classifier, which can integrate multiple effective features for classifying Apps. Actually, for this problem, a lot of supervised classification models, such as Naive Bayes, SVMs, Decision Tree and Maximum Entropy (MaxEnt) can be taken advantage of in our framework for App classification. Among them, in this paper we propose to leverage MaxEnt for training a mobile App classifier due to three major reasons [23], [24]: 1) MaxEnt is robust and successfully applied to a wide range of NLP tasks, such as POS tagging, and other classification problems. It is proven to perform better than other alternative models in classifying insufficient and sparse data. 2) Compared with other classification approaches, MaxEnt is more flexible to incorporate different types of features, such as the various features extracted from a Web search engine and real-world context logs. 3) MaxEnt is very efficient in processes of both training and testing, which is suitable for deployment on mobile devices.

In our problem, MaxEnt defines the conditional probability of a category label $c$ given an observation App name $a$ as:

$$P(c|a) = \frac{1}{Z(a)} \exp(\sum_i \lambda_i f_i(a,c)), \tag{12}$$

where $Z(a) = \sum_c \exp(\sum_i \lambda_i f_i(a,c))$ is a normalization factor, each $f_i(a,c)$ denotes a feature function, and $\lambda_i$ indicates the weight of $f_i(a,c)$. Given a training data set $\mathcal{D} = \{a^{(i)}, c^{(i)}\}_{i=1}^N$, the objective of training a MaxEnt model is to find a set of parameters $\Lambda = \{\lambda_i\}$ that maximize the conditional log-likelihood:

$$L(\Lambda|\mathcal{D}) = \log \prod_{d \in \mathcal{D}} P_\Lambda(c^{(i)}|a^{(i)}). \tag{13}$$

To be specific, we can leverage many machine learning algorithms to train MaxEnt model, such as Improved Iterative Scaling (IIS) [13] and Limited-Memory BFGS (L-BFGS) [21]. In this paper, according to the comparison results of algorithms for maximum entropy parameter estimation in [21], we leverage the most efficient algorithm L-BFGS for model training. Once the parameters $\Lambda$ have been learned by using a training data set, we can infer the category label $c_T^*$ for the test App $a_T$ as $c_T^* = \arg\max_{c_T} P(c_T|a_T, \Lambda)$.

## 5 Experimental Results

In this section, we evaluate our approach through systematic empirical comparisons with two state-of-the-art baselines on a real-world data set.

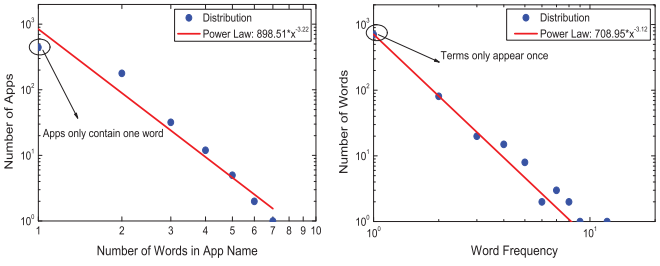
TABLE 4
The types of contextual information in our data set.

| Context | Value range |
|---|---|
| Day name | {Monday, Tuesday,... , Sunday} |
| Is a holiday? | {Yes, No} |
| Day period | {Morning(7:00-11:00), Noon(11:00-14:00), Afternoon(14:00-18:00), Evening(18:00-21:00), Night(21:00-Next day 7:00)} |
| Time range | {0:00-1:00, 1:00-2:00, ... , 23:00-24:00} |
| Profile | {General, Silent, Meeting, Outdoor, Pager, Offline} |
| Battery level | {Level 1, Level 2, ... , Level 7} |
| Charging state | {Charging, Complete, Not Connected} |
| Location | {Home, Work Place, On the Way}. |

### 5.1 Experimental Set Up and Data Set

The data set used in the experiments is collected from 443 volunteers by a major manufacturer of smart mobile devices (i.e., Nokia Corporation), during the period of 2007 to 2008 in UK. Specifically, all the volunteers were requested to install a data collection client in their Nokia S60 smart phones. The client can run in background and collect rich context data such as GPS data, system information, GSM data, sensor data, and App usage records, with fixed sampling rate. For each mobile device, the client software automatically uploads the collected data to the server through the GPRS/Wi-Fi Internet. In the server, context logs are built from the collected context data and interaction records for each volunteer. In this data set, all these 443 users used 680 unique mobile Apps, and their context logs contain total 8,852,187 context records spanning for from several weeks to several months. Some similar public data sets can be found in [1], [2].

Table 4 shows the concrete types of context data the data set contains. Figure 7 shows the distribution of the number of mobile Apps with respect to the name length and the distribution of the number of unique words in App names with respect to their appearing frequency in our real-world date set, which clearly validates the sparseness of textual information in App names.




(a) (b)

Fig. 7. The distribution of (a) the number of mobile Apps with respect to the name length, and (b) the number of unique words in App names with respect to their appearing frequency in our real-world date set.

In the experiments, we manually define a two-level App taxonomy based on the taxonomy of Nokia Store, which contains 9 level-1 categories and 27 level-2 categories. Table 5 shows the details of our App taxonomy.

We invited three human labelers who are familiar with smart mobile devices and Apps to manually label

TABLE 5
The predefined two-level taxonomy in our experiments.

| Level-1 Categories | Level-2 Categories |
|---|---|
| Internet | ⋆Web Browser, ⋆Others |
| Business | ⋆Office Tools, ⋆Security, ⋆Others |
| Communication | ⋆Call, ⋆Mail&SMS, ⋆Others |
| Game | ⋆Action, ⋆Strategy, ⋆Others |
| Multimedia | ⋆Audio, ⋆Video, ⋆Others |
| Navigation | ⋆City Guides, ⋆Maps, ⋆Others |
| SNS | ⋆IM, ⋆Blog&Forum, ⋆Others |
| System | ⋆Management, ⋆Performance, ⋆Others |
| Reference | ⋆News, ⋆Utility, ⋆Reading, ⋆Others |

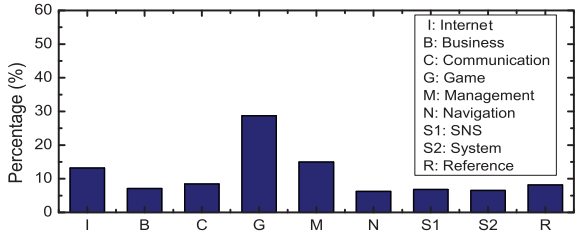



Fig. 8. The App distribution of different level-1 category labels in our data set.

the total 680 Apps with the 27 level-2 category labels. For each App, each labeler gave the most appropriate category label by his (or her) own usage experience (all the Apps in the experiments can be downloaded through Nokia Store). The final label of each App was voted by three labelers. Particularly, for more than 95% apps, the three labelers gave the same labels. Figure 8 shows the category distribution of the labeled Apps. From this figure, we can observe that the category labels of the Apps in our data set cover all nine level-1 categories and the distribution is relatively even.

### 5.2 Benchmark Methods

In this paper, we adopt two state-of-the-art baselines to evaluate the performance of our classification approach. To the best of our knowledge, there is only one relevant approach has been reported in recent years, which can be directly leveraged for automatic App classification. Therefore, we leverage this approach as the first baseline.

**Word Vector based App Classifier** (WVAC) is introduced in [20], which is adapted from the Web query classification approach proposed by Cao [11] for App usage record normalization. To be specific, given an App $a$, it directly calculates the Cosine similarity between category word vector $\overrightarrow{w_c}$ and App word vector $\overrightarrow{w_a}$, and label $a$ with category $c^*$ i.i.f. $c^* = \arg\max_c Similarity(\overrightarrow{w_c}, \overrightarrow{w_a})$.

The second baseline is originally developed for short & sparse text classification, which can be extended for classifying Apps.

**Hidden Topic based App Classification** (HTAC) is introduced in [23], whose main idea is to learn hidden topics for enriching original short & sparse texts. To be specific, this approach adds semantic topics as additional textual features and integrate them with words for classifying short & sparse texts. To leverage this approach for classifying mobile Apps, we first extract the semantic topics by the approach introduced in Section 4.1.2, and then combine them with the words in App names for training a MaxEnt classifier.

### 5.3 Evaluation Metrics

To reduce the uncertainty of splitting the data into training and test data, in the experiments we utilize ten-fold cross validation to evaluate each classification approach. To be specific, we first randomly divide 680 Apps into ten equal parts, and then use each part as the test data while using other nine parts as the training data in ten test rounds. Finally, we report the average performance of each approach in the ten rounds of tests. To evaluate the classification performance of each approach, we leverage three metrics as follows.

**Overall Precision@K** is calculated by $\frac{\sum_{n=1}^{N} P@K_n}{N}$, where $N$ indicates the number of apps in the test data set and $P@K_n$ indicates the precision for the $n$-th test App with a set of top $K$ predicted category labels $C_K$ from a classification approach. To be specific, $P@K = \frac{\delta(c^* \in C_K)}{|K|}$, where $c^*$ denotes the ground truth of category label for a test App, and $\delta(*)$ denotes a boolean function of indicating whether $*$ is true ($\delta(*) = 1$) or false ($\delta(*) = 0$).

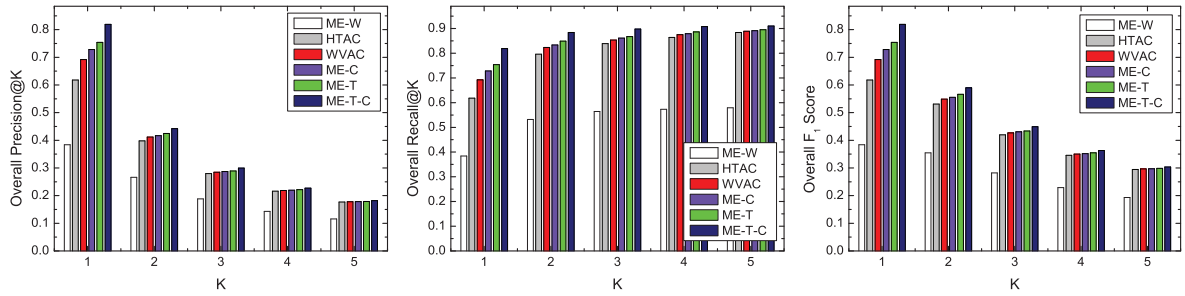
**Overall Recall@K** is calculated by $\frac{\sum_{n=1}^{N} R@K_n}{N}$, where $R@K_n$ denotes the recall for the $n$-th test App with a set of top $K$ predicted category labels $C_K$ from a classification approach. To be specific, $R@K = \delta(c^* \in C_K)$.

**Overall $F_1$ Score** is calculated by $\frac{\sum_{n=1}^{N} F@K_n}{N}$, where $F@K_n$ denotes the $F_1$ score for the $n$-th test App with a set of top $K$ predicted category labels $C_K$ from a classification approach. To be specific, $F@K = \frac{2 \times P@K \times R@K}{P@K + R@K}$.

### 5.4 Overall Results and Analysis

In order to study the contribution of Web knowledge based textual features and contextual features in our approach, we compare four MaxEnt models with different features, namely, ME-W (*MaxEnt with Words*), ME-T (*MaxEnt with words + Web knowledge based Textual features*), ME-C (*MaxEnt with words + Contextual Features*) and ME-T-C (*MaxEnt with words + Web knowledge based Textual features + Contextual Features*). Because we treat the words in App names as basic features, all models take advantage of this kind of features by default.

In our experiments, we choose Google as our Web search engine to obtain the relevant snippets of Apps, and set the number of search results $M$ to be 10, which equals to the number of search results in one search page. Each search snippet is normalized by Stop-Words Remover [4] and Porter Stemmer [5]. The number of latent topic $K$ for both our approach and baseline HTAC are set to 20 according to the estimation approach introduced in [6], [33]. Two parameters $\alpha$ and $\beta$ for training LDA model are set to be $50/K$ and 0.1 according to [16]. The parameters for training LDAC model are set as similar as [6]. The settings of context pattern mining approach BP-Growth are similar to [18]. To avoid over fitting in the training process of MaxEnt model,




(a) Overall Precision@K (b) Overall Recall@K (c) Overall $F_1$ Score

Fig. 9. The overall performance of each classification approach with different evaluation metrics in the cross validation.

TABLE 6
The mean deviations of $Precision@K$ of each classification approach with different $K$ in the ten-fold cross validation.

| | P@1 | P@2 | P@3 | P@4 | P@5 |
|---|---|---|---|---|---|
| ME-W | 0.4757 | 0.2479 | 0.1643 | 0.1234 | 0.0973 |
| HTAC | 0.4432 | 0.1569 | 0.0912 | 0.0576 | 0.0422 |
| WVAC | 0.4322 | 0.1436 | 0.0791 | 0.0577 | 0.0387 |
| ME-C | 0.3914 | 0.1339 | 0.0849 | 0.0562 | 0.0369 |
| ME-T | 0.3655 | 0.1314 | 0.0764 | 0.0526 | 0.0361 |
| ME-T-C | **0.2955** | **0.1009** | **0.0625** | **0.0422** | **0.0334** |

TABLE 7
The mean deviations of $Recall@K$ of each classification approach with different $K$ in the ten-fold cross validation.

| | R@1 | R@2 | R@3 | R@4 | R@5 |
|---|---|---|---|---|---|
| ME-W | 0.4757 | 0.4959 | 0.4930 | 0.4936 | 0.4863 |
| HTAC | 0.4432 | 0.3137 | 0.2737 | 0.2302 | 0.2109 |
| WVAC | 0.4322 | 0.2872 | 0.2373 | 0.2309 | 0.1934 |
| ME-C | 0.3914 | 0.2679 | 0.2547 | 0.2247 | 0.1849 |
| ME-T | 0.3655 | 0.2629 | 0.2291 | 0.2105 | 0.1805 |
| ME-T-C | **0.2955** | **0.2018** | **0.1875** | **0.1688** | **0.1672** |

TABLE 8
The mean deviations of $F_1$ score of each classification approach with different $K$ in the ten-fold cross validation.

| | F@1 | F@2 | F@3 | F@4 | F@5 |
|---|---|---|---|---|---|
| ME-W | 0.4757 | 0.3306 | 0.2465 | 0.1974 | 0.1621 |
| HTAC | 0.4432 | 0.2091 | 0.1368 | 0.0921 | 0.0703 |
| WVAC | 0.4322 | 0.1915 | 0.1187 | 0.0924 | 0.0645 |
| ME-C | 0.3914 | 0.1786 | 0.1274 | 0.0899 | 0.0616 |
| ME-T | 0.3655 | 0.1753 | 0.1145 | 0.0842 | 0.0602 |
| ME-T-C | **0.2955** | **0.1346** | **0.0937** | **0.0675** | **0.0557** |

we also use Gaussian prior for parameter $\Lambda$ as similar as [22]. Moreover, both our approach and the baselines are implemented by standard C++ and the experiments are conducted on a 3GHZ×4 quad-core CPU, 3G main memory PC. Here, we evaluate the overall $Precision@K$, overall $Recall@K$ and overall $F_1$ score with different $K$ for each classification approach. To be specific, we set the maximum $K$ to be 5.

Figure 9 (a) compares the average overall $Precision@K$ of two baseline methods WVAC, HTAC and our approach with different features, namely, ME-W, ME-T, ME-C and ME-T-C in the ten rounds of tests. First, from the figure we can observe that the classification performance of only leveraging the short & sparse texts in App names (i.e., ME-W) is very limited. Second, compared with the two baselines WVAC and HTAC, the average overall $Precision@K$ of our approaches ME-T, ME-C and ME-T-C is improved consistently. To be specific, for the top 1 results (i.e., given $K = 1$), the improvement is more than 9% (ME-T), 6% (ME-C) and 19% (ME-T-C) with respect to WVAC, and 22%, 19% and 34% with respect to HTAC. Third, comparing ME-T and ME-C, we can observe that the Web knowledge based textual features are slightly more effective than contextual features though both of them effectively improve the performance of App classification than ME-W, which only leverages the words in App names. Last, ME-T-C outperforms all other approaches in terms of average overall $Precision@K$. The average improvement than ME-W across different $K$ is more than 70% (the improvement exceeds 110% given $K = 1$), which clearly validates our motivation of leveraging both Web knowledge based textual features and real-world contextual features for improving the performance of App classification.

Similarly, Figure 9 (b) compares the average overall $Recall@K$ of ME-W, ME-T, ME-C, ME-T-C and two baselines with respect to different $K$ in the ten rounds of tests. From this figure we can observe that our approaches outperform the baselines and ME-T-C has the best performance. Another observation is that the average overall $Recall@K$ of each test approach increases with the increase of $K$, which is reasonable because the probability that the ground truth category label is covered by the predicted results will increase with more predicted category labels. Moreover, Figure 9 (c) compares the average overall $F_1$ score of all test approaches in the ten rounds of tests. From this figure we can observe that ME-T-C consistently outperforms other approaches and ME-W has the worst classification performance in terms of $F_1$ score.

Particularly, we conduct a series of paired T-test of 0.95 confidence level which show that the improvements of our approaches ME-T-C on overall $Precision@K$, overall $Recall@K$ and overall $F_1$ score with different $K$ to other approaches are all statistically significant.

We also study the variances of overall $Precision@K$, overall $Recall@K$ and overall $F_1$ score of all test approaches in the ten-fold cross validation with $K \in [1, 5]$. Table 6, Table 7, and Table 8 show the mean deviations of these metric values of each approach in the ten rounds of tests. From these tables we can observe that the variances of all other approaches are consistently smaller than ME-W, which implies that taking advantage of additional

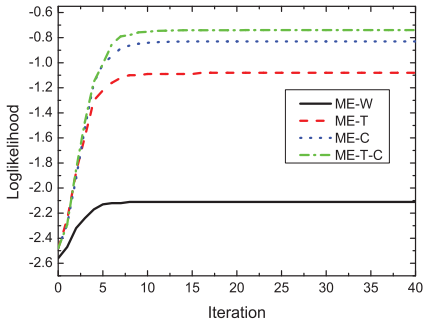



Fig. 10. The objective function values per iteration of training ME-W, ME-T, ME-C and ME-T-C.

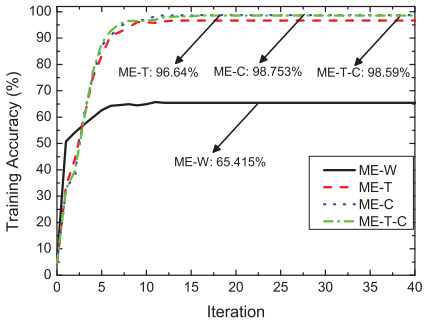



Fig. 11. The values of training accuracy per iteration of training ME-W, ME-T, ME-C and ME-T-C.

features other than the limited textual information in App names can improve the robustness of App classification. Moreover, ME-T-C has the smallest mean deviations on all metrics with different $K$, which implies that it has the best robustness among all test approaches.

From the above experiments, we can draw the conclusions as follows: 1) All other approaches outperform ME-W, which implies the textual information in App names is insufficient for classifying Apps effectively and leveraging additional features can improve the classification performance dramatically. 2) The MaxEnt model with Web knowledge based textual features, i.e., ME-T, outperforms the two baselines WVAC and HTAC, which indicates that the combination of multiple Web knowledge based textual features and basic App name based features is more effective than single Web knowledge based textual features for App classification. 3) The MaxEnt model with contextual features ME-C also outperforms two baselines, which validates the effectiveness of relevant contexts for improving the App classification performance. 4) The MaxEnt model which combines the Web knowledge based textual features and real-world contextual features, i.e., ME-T-C, outperforms both ME-T and ME-C, which indicates the integration of two kinds of additional features in the MaxEnt model can achieve the best performance.

### 5.5 The Efficiency of Our Approach

Our approach consists of an offline part and an online part. In the offline part, the time cost of our approach majorally comes from the training cost for the MaxEnt model. Figure 10 shows the convergence curves of ME-W, ME-T, ME-C and ME-T-C by measuring their log likelihood for the training data set in one of the ten test rounds. From these figures we can observe that the L-BFGS training of all approaches converges quickly. We can also find that the objective function value of ME-T-C converges to a better optima compared to other approaches, and the objective function value of ME-W converges to the worst optima point compared with other approaches. The convergence curves for other test rounds follow the similar trend. Moreover, each iteration of L-BFGS training averagely costs 2.8 milliseconds for ME-W, 8 milliseconds for ME-T, 15 milliseconds for ME-C and 18 milliseconds for ME-T-C, respectively. We also show the curves of *training accuracy* with respect to training iterations of all above approaches in Figure 11, where training accuracy denotes the classification accuracy of the trained model on the training data. Similarly to the curves of log likelihood, we can observe that the training accuracy curves of all approaches converge quickly and ME-T, ME-C and ME-T-C can achieve a high training accuracy while ME-W can only achieve about 65% training accuracy at best.

In the online part, we need to submit App names to a Web search engine for getting the relevant snippets. Indeed, this process can be very fast for a commercial search engine thus it is not a crucial efficiency problem. The other online cost of our approach comes from feature generation, such as calculating label/topic confidence, calculating category rank distance and mining context patterns. Actually, the main processes of these tasks can be calculated offline in advance. To be specific, both VSM and LDA/LDAC models can be trained offline and the context patterns can also be mined in advance and stored in the server. In this case, the online process for generating features will be very fast (less than 100 millisecond in our experiments).

### 5.6 Case Study of App Classification

In addition to the studies on the overall performance of all test approaches, we also manually study the case in which our approach outperforms the baselines.

For example, Table 9 shows an example of the App classification results of different test approaches with respect to different features. In this example, the test App is "*Snake 3D*", which is a popular action games thus the ground truth category label is "*Game/Action Game*". From the results we can observe that the approach which only leverages the words in App name, i.e., ME-W, cannot give the correct label in the top three results. The two baselines (i.e., WVAC and HTAC) gave the correct label in the third position. Moreover, the Web knowledge based textual feature and contextual feature based approaches (i.e., ME-T and ME-C) gave the correct label in the second position. In contrast, our approach which combines both Web knowledge based textual features and contextual features (i.e., ME-T-C) gave the correct label in the top one position.

We can have some interesting insights from this case. Specifically, first, the Web knowledge based textual features, i.e., the relevant snippets and topics, can reflect

TABLE 9
A case study of the App classification results of different approaches.

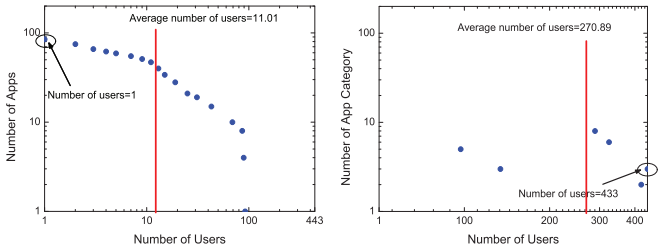
| App Name | Snakes 3D |
|---|---|
| Ground Truth | Game/Action |
| Words | Snakes, 3D |
| Snippets | →**Snake 3d - Free Online Games (FOG)** www.freeonlinegames.com/game/snake-3d →**3D Snake - squid soup : games** squidsoup.com/games/snake/shockholder.html →**Snake 3D - 2 Flash Games** www.2flashgames.com |
| Topics† | Entertainment, Video Games, Mobile Applications |
| Context Topics† | Relax at Home, Relax at Work Place, After Work |
| Context Patt.s | {(*Is a holiday?: Yes*)(*Day period: Evening*) (*Location: Home*)}; {(*Day period: Evening*)(*Charging state: Charging*) (*Location: Home*)}; {(*Day period: Afternoon*) (*Profile: Silent*)} |
| Predicted Category Labels | |
| WVAC | Multimedia/Video; Game/Others; **Game/Action** |
| HTAC | Multimedia/Video; Internet/Others; **Game/Action** |
| ME-W | Multimedia/Video; Multimedia/Others; Game/Others |
| ME-T | Game/Others; **Game/Action**; Multimedia/Video |
| ME-C | Multimedia/Video; **Game/Action**; Game/Others |
| ME-T-C | **Game/Action**; Multimedia/Video; Game/Others |

* Limited to space, we only show top three corresponding results.
† The topics are manually labeled for illustration.

that the App is probably a game but cannot determine whether it is an action game. By further considering the relevant context patterns, we can find that it is usually played in a quiet and relaxing environment with long time to play (in holiday evenings at home with silent profile), which implies it is probably an action game since such environment eases users to finish action tasks. Thus, by considering both types of features, ME-T-C made a more accurate classification result for the App.

### 5.7 App Usage based User Segmentation

Except for directly evaluating the classification performance of our approach (i.e. ME-T-C), we also study its effectiveness of segmenting users with respect to their App usage. Specifically, the user segmentation aims to cluster users according to their similarities of App usage, which can motivate many useful services, such as App recommendation and user habit analysis. However, it is not a trivial work to effectively segment users with respect to their historical App usage, since the original App distribution is very sparse in users' App usage records. For example, Figure 12 (a) shows the distribution of the number of used Apps with respect to number of users in our data set. From this figure we can observe that the distribution is very sparse, and each App only have 11 users on average. If we leverage these Apps for measuring user similarity, it will be very hard to obtain good segmentation performance. Fortunately, if we map each original App to a predefined category




(a) (b)

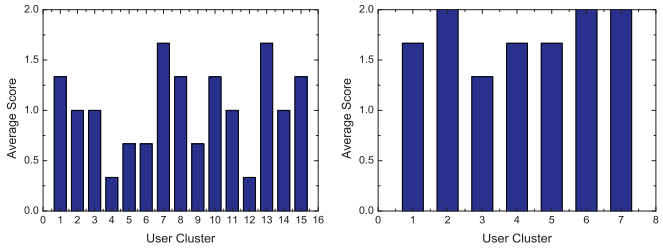
Fig. 12. The distributions of the number of used (a) Apps and (b) categories with respect to the number of users.

labels, the sparsity can be reduced. Figure 12 (b) shows the distribution of the number of used App categories (i.e., classified by ME-T-C) with respect to the number of users in our data set. From this figure we can find that the distribution is relatively even, and each App category have 270 users on average. Intuitively, using these category labels for user segmentation may obtain better performance.

Specifically, in this sub-section we first cluster the users according to their similarities of App usage which are calculated by the Cosine similarities between their original App vectors and App category vectors, respectively, and compare their performance. The original App vector of user $u$ is denoted as $\overrightarrow{u_a} = dim[n]$ , where $n = 680$ is the number of all unique original Apps. $dim[i] = \frac{freq_{i,u}}{\sum_i freq_{i,u}}$, where $freq_{i,u}$ is the frequency of $i$-th App in $u$'s historical context log. Similarly, the App category vector of user $u$ is denoted as $\overrightarrow{u_c}$. To efficiently segment users, we utilize a clustering algorithm proposed in [12], which does not require a parameter to indicate the number of clusters but only needs a parameter to indicate the minimum average mutual similarity $S_{min}$ for the data points in each cluster. The average mutual similarity for a user cluster $C$ is calculated as $S_C = \frac{2\times\sum_{1\leq i<j\leq N_C} Sim(u_i,u_j)}{N_C\times(N_C-1)}$, where $N_C$ indicates the number of users in $C$ and $Sim(u_i, u_j)$ denotes the Cosine similarity between the $i$-th user and the $j$-th user in $C$.

For the clusters based on App category vectors, $S_{min}$ is empirically set to be 0.8. However, for the clusters based on original App vectors, it is difficult to select a proper $S_{min}$ because there exist rare pairs of users whose similarities are relatively big when the similarity is calculated based on the sparse original App space. Through several trials, we choose $S_{min} = 0.3$ for those clusters since in this case the results look relatively good. After clustering, the clusters number of App category vectors and original App vectors based approaches are 7 and 15, respectively.

To evaluate the segmentation performance, we leverage the provided questionnaires in the data set. The questionnaires are filled by the 443 volunteers, which contain serval App usage related questions, which can indicate the user preferences accurately. The main questions include *Which (kinds of) Apps do you use most frequently?*, *Which (kinds of) Apps do you most interested in?* and *Which (kinds of) Apps do you dislike most?*. In this experiment, we also invite three human evaluators




(a) (b)

Fig. 13. The segmentation performance based on (a) original App vectors and (b) App category vectors.

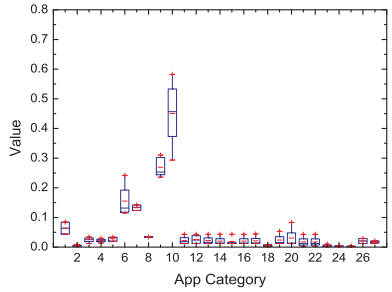



Fig. 14. The values of each App category dimension for the users in $C_5$ with box plots.

to judge the segmentation performance. Specifically, for each user cluster $C$, the three evaluators should first read the questionnaires filled by users belong to $C$, and judge the segmentation performance with a score from 0 (worst) to 2 (best) according to their own perspective. At last, we use the average score for judging the performance of user segmentation.

Figure 13 (a) and (b) show the evaluation results for each user cluster mined based on original App vectors and App category vectors, respectively. From these figures we can observe that the App category vector based segmentation outperforms the original App vectors based segmentation, which is because the App categories can reduce the sparsity of original App space and thus can capture the user similarity much better. It also indicates the effectiveness of our approach ME-T-C.

We also study the user clusters based on App categories and find all of them have obvious relevance with particular App category preferences. For example, Figure 14 shows the values of each App category dimension for the users in one randomly selected clusters $C_5$ with box plots. From this figure we can clearly see that the users in the cluster dramatically have high values in some App category dimensions, which implies they have the preferences of the corresponding App categories. To be specific, the users in $C_5$ seem to like the 9-th and 10-th App categories which indicate the categories "*Game/Action Game*" and "*Game/Strategy Game*".

## 6 Concluding Remarks

In this paper, we studied the problem of automatic App classification. A critical problem along this line is the contextual information in App names is insufficient and sparse for achieving a good classification performance. To this end, we proposed a novel approach for classifying mobile Apps by leveraging both Web knowledge and relevant real-world context. To be specific, we first extracted several Web knowledge based textual features by taking advantage of a Web search engine. Then, we also leveraged real-world context logs which record the usage of Apps and corresponding contexts to extract relevant contextual features. Finally, we integrated both types of features into a widely used MaxEnt model for training an App classifier. The experiments on a real-world data set collected from 443 mobile users clearly show that our approach outperforms two state-of-the-arts baselines.

Although our current approach is both efficient and effective for solving the problem of automatic App classification, it is still an open problem about how to embed this approach into mobile devices. Since mobile devices have very limited computing resources, it is necessary to design a more effective service framework. Moreover, different users may have different App usage behaviors, thus how to integrate such personal preferences into contextual feature extraction will be an interesting research direction. Finally, in our future research, we also plan to combine our classification approach with other context-ware services, such as context-aware App recommender system, to enhance user experiences.

**Acknowledgement.** This work was supported in part by grants from Natural Science Foundation of China (NSFC, Grant No. 61073110 and 71028002), Research Fund for the Doctoral Program of Higher Education of China (Grant No. 20113402110024), the Key Program of National Natural Science Foundation of China (Grant No. 60933013), and National Key Technology Research and Development Program of the Ministry of Science and Technology of China (Grant No. 2012BAH17B03). This work was also partially supported by grants from National Science Foundation (NSF) via grant numbers CCF-1018151 and IIS-1256016.

## References

[1] http://realitycommons.media.mit.edu/realitymining.html.
[2] http://research.nokia.com/page/12000.
[3] http://store.ovi.com/.
[4] http://www.lextek.com/manuals/onix/index.html.
[5] http://www.ling.gu.se/lager/mogul/porter-stemmer.
[6] T. Bao, H. Cao, E. Chen, J. Tian, and H. Xiong. An unsupervised approach to modeling personalized contexts of mobile users. In *ICDM'10*, pages 38–47, 2010.
[7] A. L. Berger, V. J. D. Pietra, and S. A. D. Pietra. A maximum entropy approach to natural language processing. *Comput. Linguist.*, 22:39–71, 1996.
[8] D. M. Blei, A. Y. Ng, and M. I. Jordan. Lantent dirichlet allocation. *Journal of Machine Learning Research*, pages 993–1022, 2003.
[9] A. Z. Broder, M. Fontoura, E. Gabrilovich, A. Joshi, V. Josifovski, and T. Zhang. Robust classification of rare queries using web knowledge. In *SIGIR '07*, pages 231–238, 2007.
[10] H. Cao, T. Bao, Q. Yang, E. Chen, and J. Tian. An effective approach for mining mobile user habits. In *CIKM'10*, pages 1677–1680, 2010.
[11] H. Cao, D. H. Hu, D. Shen, D. Jiang, J.-T. Sun, E. Chen, and Q. Yang. Context-aware query classification. In *SIGIR'09*, pages 3–10, 2009.
[12] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li. Context-aware query suggestion by mining click-through and session data. In *KDD'08*, pages 875–883, 2008.

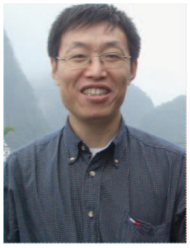
[13] S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:380–393, 1997.
[14] Y. Ge, Q. Liu, H. Xiong, A. Tuzhilin, and J. Chen. Cost-aware travel tour recommendation. In *KDD'11*, pages 983–991, 2011.
[15] T. L. Griffiths and M. Steyvers. Finding scientific topics. *In Proc. of National Academy of Science of the USA*, pages 5228–5235, 2004.
[16] G. Heinrich. Paramter stimaion for text analysis. *Technical report, University of Lipzig*, 2008.
[17] M. Kahng, S. Lee, and S.-g. Lee. Ranking in context-aware recommender systems. In *WWW'11*, pages 65–66, 2011.
[18] X. Li, H. Cao, H. Xiong, E. Chen, and J. Tian. Bp-growth: Searching strategies for efficient behavior pattern mining. In *MDM'12*, pages 238–247, 2012.
[19] Q. Liu, Y. Ge, Z. Li, E. Chen, and H. Xiong. Personalized travel package recommendation. In *ICDM'11*, pages 407 – 416, 2011.
[20] H. Ma, H. Cao, Q. Yang, E. Chen, and J. Tian. A habit mining approach for discovering similar mobile users. In *WWW'12*, pages 231–240, 2012.
[21] R. Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *COLING-02*, pages 1–7, 2002.
[22] K. Nigam. Using maximum entropy for text classification. In *In IJCAI-99 Workshop on Machine Learning for Information Filtering*, pages 61–67, 1999.
[23] X.-H. Phan, C.-T. Nguyen, D.-T. Le, L.-M. Nguyen, S. Horiguchi, and Q.-T. Ha. A hidden topic-based framework toward building applications with short web documents. *IEEE Transactions on Knowledge and Data Engineering*, 23:961 – 976, 2010.
[24] X.-H. Phan, L.-M. Nguyen, and S. Horiguchi. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *WWW '08*, pages 91–100, 2008.
[25] M. Sahami and T. D. Heilman. A web-based kernel function for measuring the similarity of short text snippets. In *WWW '06*, pages 377–386, 2006.
[26] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18:613–620, 1975.
[27] D. Shen, J.-T. Sun, Q. Yang, and Z. Chen. Building bridges for web query classification. In *SIGIR '06*, pages 131–138, 2006.
[28] M. van Setten, S. Pokraev, and J. Koolwaaij. Context-aware recommendations in the mobile tourist application compass. In *AH'2004*, pages 235–244, 2004.
[29] W. Woerndl, C. Schueller, and R. Wojtech. A hybrid recommender system for context-aware recommendations of mobile applications. In *ICDE'07*, pages 871–878, 2007.
[30] W.-T. Yih and C. Meek. Improving similarity measures for short segments of text. In *Proceedings of the 22nd national conference on Artificial intelligence - Volume 2*, pages 1489–1494, 2007.
[31] K. Yu, B. Zhang, H. Zhu, H. Cao, and J. Tian. Towards personalized context-aware recommendation by mining context logs through topic models. In *PAKDD'12*, pages 431–443, 2012.
[32] H. Zhu, H. Cao, E. Chen, H. Xiong, and J. Tian. Exploiting enriched contextual information for mobile app classification. In *CIKM'12*, pages 1617–1621, 2012.
[33] H. Zhu, H. Cao, H. Xiong, E. Chen, and J. Tian. Towards expert finding by leveraging relevant categories in authority ranking. In *CIKM '11*, pages 2221–2224, 2011.
[34] H. Zhu, E. Chen, K. Yu, H. Cao, H. Xiong, and J. Tian. Mining personal context-aware preferences for mobile users. In *ICDM'12*, pages 1212–1217, 2012.

**Hengshu Zhu** is currently a Ph.D. student in the School of Computer Science and Technology at University of Science and Technology of China (USTC). He was supported by the China Scholarship Council as a visiting research student at Rutgers, the State University of New Jersey, for more than one year. He received his B.E. degree in Computer Science in 2009 from USTC.

His main research interests include mobile data mining, recommender systems, and social networks. During his Ph.D. study, he has published a number of papers in refereed conference proceedings and journals, such as CIKM, ICDM and WWW Journal. Two of his papers were awarded as "the Best Student Paper" of KSEM'11 and WAIM'13, respectively. He has also been a journal reviewer for TSMC-B, WWW Journal and KAIS, and an external reviewer for various international conferences, such as KDD and ICDM.

**Huanhuan Cao** is currently the principal scientist of Nuomi.com. Before joining Nuomi, he used to be a key resarch engineer of Hipu Info. Tech. Ltd, which is a start-up focusing on personalized news recommendation. Earlier, he worked in Nokia Research Center as a senior researcher. During his student time, he received a B.E. degree and a Ph.D. degree from University of Science and Technology of China, China, in 2005 and 2009, respectively. Due to his academic achievement of his Ph.D research work, he won the Microsoft Fellow and Chinese Academic Science President Award.

In recent years, his major research interests included recommender system, location mining, and mobile user behavior analysis. In these fields, he has applied more than 30 invention patents and published more than 20 papers in high rated conferences and journals.



**Enhong Chen** (SM'07) is a Professor and vice dean of School of Computer Science and Technology at University of Science and Technology of China (USTC), China. He received the Ph.D. degree from the University of Science and Technology of China, China.

His general area of research is data mining, personalized recommendation systems and web information processing. He has published more than 100 papers in refereed conferences and journals. His research is supported by the National Natural Science Foundation of China, National High Technology Research and Development Program 863 of China, etc. He is the program committee member of more than 30 international conferences and workshops. He is a senior member of the IEEE.

**Hui Xiong** is currently an Associate Professor and Vice Chair of the Management Science and Information Systems Department, and the Director of Rutgers Center for Information Assurance at the Rutgers, the State University of New Jersey, where he received a two-year early promotion/tenure (2009), the Rutgers University Board of Trustees Research Fellowship for Scholarly Excellence (2009), and the ICDM-2011 Best Research Paper Award (2011). He received the B.E. degree from the University of Science and Technology of China (USTC), China, the M.S. degree from the National University of Singapore (NUS), Singapore, and the Ph.D. degree from the University of Minnesota (UMN), USA.

His general area of research is data and knowledge engineering, with a focus on developing effective and efficient data analysis techniques for emerging data intensive applications. He has published prolifically in refereed journals and conference proceedings (3 books, 40+ journal papers, and 60+ conference papers). He is a co-Editor-in-Chief of Encyclopedia of GIS, an Associate Editor of IEEE Transactions on Data and Knowledge Engineering (TKDE) and the Knowledge and Information Systems (KAIS) journal. He has served regularly on the organization and program committees of numerous conferences, including as a Program Co-Chair of the Industrial and Government Track for the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining and a Program Co-Chair for the 2013 IEEE International Conference on Data Mining (ICDM-2013). He is a senior member of the ACM and IEEE.

**Jilei Tian** received his B.S., M.S. degrees in Biomedical Engineering from Xi'an Jiaotong University, China and Ph.D. degree in Computer Science from University of Eastern Finland, in 1985, 1988 and 1997, respectively. He joined Beijing Jiaotong University faculty during 1988-1994. He joined Nokia Research Center as senior researcher since 1997, then as principal scientist and research leader, primarily in the area of spoken language processing and recently on rich context data modeling and personalized services. He has authored more than 100 publications including book chapter, journal and conference papers. He has also about 100 patents including pending. He has served as member of technical committee and the editorial board of international conferences and journals.