Exploiting Enriched Contextual Information for Mobile App Classification

Hengshu Zhu[1] Huanhuan Cao[2] Enhong Chen[1] Hui Xiong[3] Jilei Tian[2]
[1]University of Science and Technology of China [2]Nokia Research Center [3]Rutgers University
[1]{zhs, cheneh}@ustc.edu.cn [2]{happia.cao, jilei.tian}@nokia.com [3]hxiong@rutgers.edu

ABSTRACT

A key step for the mobile app usage analysis is to classify apps into some predefined categories. However, it is a non-trivial task to effectively classify mobile apps due to the limited contextual information available for the analysis. To this end, in this paper, we propose an approach to first enrich 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 a Maximum Entropy model for training a mobile app classifier. The experimental results based on 443 mobile users' device logs clearly show that our approach outperforms two state-of-the-art benchmark methods with a significant margin.

Categories and Subject Descriptors

I.5.2 [**Pattern Recognition**]: Design Methodology-*Feature evaluation and selection, Classifier design and evaluation*

Keywords

Automatic mobile app classification, Web knowledge, Real-world contexts.

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. 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 and target advertising [4, 6, 13].

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.

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. To be specific, 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.

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 [3, 9, 10, 12], an effective approach to 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. 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 [4, 6, 13] 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 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. Then, we propose to exploit the Maximum Entropy model (MaxEnt) [8] for combining the effective features to train a very effective mobile 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.

2. OVERVIEW

In this section, we first introduce several related notions and give an overview of our mobile app classification approach. Then we introduce the training model which is used in our approach for app classification.

2.1 Preliminary

App Taxonomy. To recognize the semantic meanings of apps, we can classify each app to one or more categories according a predefined *app taxonomy*. To be specific, an app taxonomy Υ 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.

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 can effectively summarize the corresponding web pages.

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 a toy 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 user 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, raw location related data in context data, such as GPS coordinates or cell IDs, have been transformed into semantic locations such as "Home" and "Work Place" by a location mining approach (e.g., [7]). The basic idea of such kinds of approach is to find the clusters of user positions and recognize their semantic meanings through a time pattern analysis.

Given a target taxonomy Υ, an app a and a system-specified parameter K, our approach incorporates the features extracted from both the relevant Web knowledge 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 $\{c_1, c_2, ..., c_{N_c}\}$ of Υ.

2.2 Training Model for App Classification

Many classification approaches, such as Naive Bayes, SVMs 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 according to the discussion in [9].

When we utilize the MaxEnt model to train app classifiers, the most important work is to select effective feature functions. 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 MaxEnt 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 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 two sections, we introduce the features extracted from both the relevant Web knowledge and contextual information for training an app classifier.

3. EXTRACTING WEB KNOWLEDGE BASED TEXTUAL FEATURES

In this section, 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 Models* and *implicit feedback of semantic topics.*

3.1 Explicit Feedback of Vector Space Models

This type of features exploits the top M results (i.e., search snippets) returned by a Web search engine by leveraging the explicit feedback of Vector Space Models (VSM) [11]. Given an app a and its category label c, we first submit a's name to a Web search engine (e.g., Google API in our experiments). Then, for each of the top M results, we map it to a category label in the app taxonomy Υ 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 verb and adjectives (e.g., "*plays* $\rightarrow$ *play*", "*better* $\rightarrow$ *good*"). Second, we build a

Table 1: A toy context log from real-world data set.

Timestamp	Context	App usage records
t_1	{(Day name: Monday),(Time range: AM8:00-9:00),(Profile: General),(Location: Home)}	Angry Birds
t_2	{(Day name: Monday),(Time range: AM8:00-9:00),(Profile: General),(Location: Moving)}	Null
t_3	{(Day name: Monday),(Time range: AM8:00-9:00),(Profile: General),(Location: Moving)}	Twitter
......		
t_{55}	{(Day name: Monday),(Time range: AM8:00-9:00),(Profile: General),(Location: Moving)}	UC Web
t_{56}	{(Day name: Monday),(Time range: AM8:00-9:00),(Profile: General),(Location: Moving)}	Null
t_{57}	{(Day name: Monday),(Time range: AM8:00-9:00),(Profile: General),(Location: Moving)}	Music Player

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, $dim[i] = \frac{freq_{i,c}}{\sum_i freq_{i,c}} (1 \leq i \leq n)$ and $freq_{i,c}$ 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. By calculating the Cosine distance between $\overrightarrow{w_{a,s}}$ and $\overrightarrow{w_c}$, we map s to the category c^* where $c^* = \arg\min_c Distance(\overrightarrow{w_{a,s}}, \overrightarrow{w_c})$.

Through the Vector Space Model, we can calculate a *general label confidence score* introduced in [3]:

$$GConf(c, a) = \frac{M_{c,a}}{M}, \tag{1}$$

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:

$$GEnt(c, a) = -\sum_{c_i \neq c} P(c_i) \log P(c_i), \tag{2}$$

where $P(c_i) = Gconf(c_i, a)$.

3.2 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 advantage of the latent semantic meaning 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 [9], the latent semantic topics can improve the performance of short&sparse text classification. Therefore, 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 model (LDA) [1] for learning latent semantic topics.

After learning latent topics, we extract the feature 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 follows:

$$D_{KL}(P(z|s)||P(z|c)) = \sum_k P(z_k|s) ln \frac{P(z_k|s)}{P(z_k|c)}, \tag{3}$$

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, i.e., $c^* = \arg\min_c D_{KL}(P(z|s)||P(z|c))$.

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

$$TConf(a, c) = \frac{T_{a,c}}{M}, \tag{4}$$

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 2.

4. EXTRACTING REAL-WORLD CONTEXTUAL FEATURES

In this section, we introduce how to extract effective contextual features of mobile apps from real-world context logs. To be specific, we study two types of contextual features, namely, *pseudo feedback from context vectors* and *frequent context patterns*.

4.1 Pseudo Feedback of Context Vectors

The first type of contextual feature 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 3.1, we can build a context vector for each app category. 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 ascending order according to their Cosine distance to $\overrightarrow{\Omega_a}$. Particularly, we define the pseudo category c^* by $c^* = \arg\min_c Distance(\overrightarrow{\Omega_a}, \overrightarrow{\Omega_c})$ and calculate the *category* *rank distance* by:

$$CRDistance(a, c) = Rk(c) - Rk(c^*) = Rk(c) - 1, \qquad (5)$$

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 Υ. Obviously, the smaller the distance, the more likely the category label c is the correct label.

4.2 Frequent Context Patterns

When leveraging the pseudo feedback of context vectors, we regard each unique context feature-value pair as an independent measure for the relevance between context 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 [2]. Along this line, we study a contextual feature to capture the relevance between the co-occurrence of contextual feature-value pairs and app usage. Specifically, we take advantage of the *frequent context patterns* for apps as follows.

It is not a trivial work to mine these context patterns. As pointed out by Cao *et al.* [2], 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. Fortunately, some researchers have studied this problem and proposed some novel algorithms for mining such context patterns. For example, Cao *et al.* [2] proposed a novel algorithm 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 [2], Li *et al* [5] proposed a more efficient algorithm named BP-Growth for 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.

After mining 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.

5. EXPERIMENTAL RESULTS

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

5.1 Experimental Set Up and Data Set

The data set used in the experiments is collected from many volunteers by a major manufacturer of smart mobile devices. The data set consists of 443 context logs from different mobile users and total 8,852,187 context records, which contains rich contextual information and the usage records of total 680 unique apps spanning for from several weeks to several months. 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 2 shows the details of our predefined app taxonomy.

Table 2: 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

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

5.2 Benchmark Methods and Evaluation Metrics

In this paper, we adopt two state-of-the-arts baselines to evaluate the performance of our 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 [7], which is adapted from the web query classification approach proposed in [3] for app usage record normalization.

The second baseline is originally developed for short text classification, which can be extended for classifying apps. **Hidden Topic based App Classification** (HTAC) is introduced in [9], 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 reduce the uncertainty of splitting the data into training and test data, in the experiments we utilize a ten-fold cross validation to evaluate each classification approach. To evaluate the classification performance of each approach, we leverage three metrics **Overall Precision@K**, **Overall Recall@K** and **Overall F_1 Score** as introduced in [3].

5.3 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 (*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

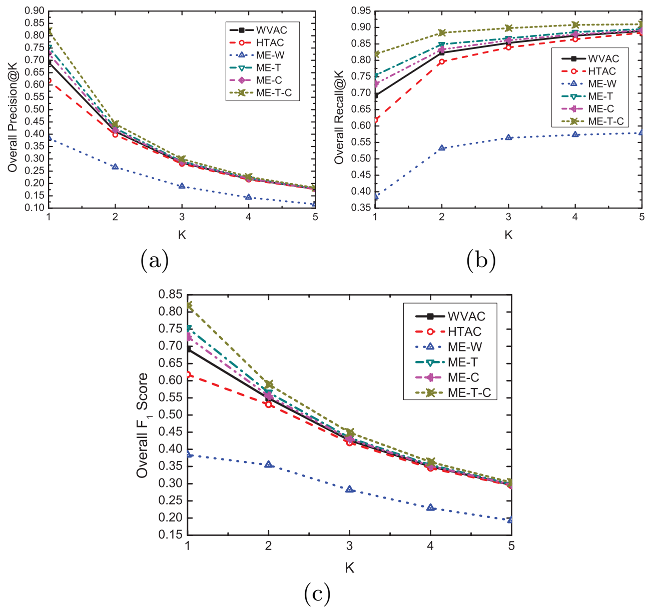

Figure 1: The average overall (a) *precision*@K, (b) *recall*@K and (c) F_1 score of each classification approach with different K in the ten-fold cross validation.

snippet is normalized by Stop-Words Remover [1] and Porter Stemmer [2]. The number of latent topic K for both our approach and baseline HTAC are set to 20 according to the estimation approach introduced in [14]. Two parameters α and β for training LDA model are set to be $50/K$ and 0.1. The settings of context pattern mining approach BP-Growth are similar to [5]. To avoid over fitting in the training process of MaxEnt model, we also use Gaussian prior for parameter Λ similar to [8]. Moreover, both our approach and the baselines are implemented by standard C++ and the experiments are conducted on a 2.8GHZ×2 Dual-core CPU, 2G main memory PC.

Figure 1 (a), (b) and (c) compares the average overall *precision*@K, *recall*@K and F_1 of two baseline methods WVAC, HTAC and our approaches with different features, namely, ME, ME-T, ME-C and ME-T-C in ten rounds of tests. From the above experiments, we can draw the conclusions as follows: 1) First, all other approaches outperform ME, 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) Second, the MaxEnt model with Web knowledge based textual features 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) Third, 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) Finally, the MaxEnt model which combines the Web knowledge based textual features and contextual features 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.

[1]http://www.lextek.com/manuals/onix/index.html
[2]http://www.ling.gu.se/lager/mogul/porter-stemmer

6. CONCLUDING REMARKS

In this paper, 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 mobile users clearly show that our approach outperforms two state-of-the-arts baselines.

Acknowledgement. This work is supported by grants from Natural Science Foundation of China (No. 61073110), Key Program of National Natural Science Foundation of China (No. 60933013), National Major Special Science & Technology Projects (No. 2011ZX04016-071), Research Fund for the Doctoral Program of Higher Education of China (No. 20093402110017) and Nokia.

7. REFERENCES

[1] D. M. Blei, A. Y. Ng, and M. I. Jordan. Lantent dirichlet allocation. *Journal of Machine Learning Research*, pages 993–1022, 2003.
[2] 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.
[3] 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.
[4] M. Kahng, S. Lee, and S.-g. Lee. Ranking in context-aware recommender systems. In *WWW'11*, pages 65–66, 2011.
[5] X. Li, H. Cao, H. Xiong, E. Chen, and J. Tian. Bp-growth: Searching strategies for efficient behavior pattern mining. In *MDM'12*, 2012.
[6] Q. Liu, Y. Ge, Z. Li, E. Chen, and H. Xiong. Personalized travel package recommendation. In *ICDM'11*, pages 407 – 416.
[7] H. Ma, H. Cao, Q. Yang, E. Chen, and J. Tian. A habit mining approach for discovering similar mobile users. In *WWW'12*, 2012.
[8] K. Nigam. Using maximum entropy for text classification. In *In IJCAI-99 Workshop on Machine Learning for Information Filtering*, pages 61–67, 1999.
[9] 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.
[10] 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.
[11] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18:613–620, 1975.
[12] D. Shen, J.-T. Sun, Q. Yang, and Z. Chen. Building bridges for web query classification. In *SIGIR '06*, pages 131–138, 2006.
[13] 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*, 2012.
[14] 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.