I.J. Information Technology and Computer Science, **2012, 10, 19-28**
Published Online September 2012 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2012.10.02

Location Based Recommendation for Mobile Users Using Language Model and Skyline Query

Qiang Pu[1,2,3]
[1]School of Information Science and Technology, Chengdu University, Chengdu, China
[2]Key Lab of Pattern Recognition & Intelligent Information Processing, Chengdu University, Chengdu, China
Email: puqiang1116@gmail.com

Ahmed Lbath[3]
[3]Joseph Fourier University of Grenoble, LIG-MRIM, Grenoble, France
Email: ahmed.lbath@imag.fr

Daqing He
School of Information Sciences of University of Pittsburgh, Pittsburgh, USA
Email: dah44@pitt.edu

***Abstract*—** Location based personalized recommendation has been introduced for the purpose of providing a mobile user with interesting information by distinguishing his preference and location. In most cases, mobile user usually does not provide all attributes of his preference or query. In extreme case, especially when mobile user is moving, he even does not provide any preference or query. Meanwhile, the recommendation system database also does not contain all attributes that can express what the user needs. In this paper, we design an effective location based recommendation system to provide the most possible interesting places to a user when he is moving, according to his implicit preference and physical moving location without the user's providing his preference or query explicitly. We proposed two circle concepts, physical position circle that represents spatial area around the user and virtual preference circle that is a non-spatial area related to user's interests. Those skyline query places in physical position circle which also match mobile user's implicit preference in virtual preference circle will be recommended. User's implicit preference will be estimated under language modeling framework according to user's historical visiting behaviors. Experiments show that our method is effective in recommending interesting places to mobile users. The main contribution of the paper comes from the combination of using skyline query and information retrieval to do an implicit location-based personalized recommendation without user's providing explicit preference or query.

***Index Terms*—** Location-Based Service, Mobile Information Recommendation, Language Model, Skyline Query, Implicit Preference

I. Introduction

Mobile-based applications have been grown as the development of mobile computing and communication technologies [1]. Since mobile devices are designed specifically for personal use, but having limited display sizes and processing power, we often need to select carefully the appropriate information to be presented to mobile users. Information recommendation for mobile users is currently an important research topic especially for location-based services using skyline query [2, 3] or spatial-temporal query [4]. Mobile information recommendation system is expected to provide more suitable and personal services to mobile user and overcome the shortages of mobile devices.

Most of location based mobile recommendation systems take into account mobile user current location as well as his preference by using skyline query [2, 3]. Skyline query retrieves a set of interesting points from a large set of objects. For example, a hotel might be a skyline interesting for a tourist, if there is no other hotel which is nearer, cheaper than this one. The tourist can choose the most promising hotel from the skyline. Here the distance and price are the attributes of hotel which user cares for.

So user's preference can be represented as a series of attributes of an object, that is also to say, users preference of interest will be affected by different dynamic attributes of the restaurant. As more attributes of an object are added into skyline query calculation, it will meet a time-consuming job [2]. Among all objects, if one or two objects are very strong compared to other objects, or when the number of category attributes is small, the resulting skyline may consist of a small number of objects [3].

Many researches focus on finding better algorithms to improve the effectiveness of skyline query, decreasing the calculation time [2, 3, 5, 6]. Because not all attributes that mobile user needs are included in database, to say nothing of the attributes that can not be presented by a value in database.

For instance, a mobile user is on his way to travel in a Chinese city in the morning. He is interested in the history of Qing dynasty, if there is a museum or temple or other places in which is exhibiting some pictures from Qing dynasty, maybe the user is more interested in those places which have nearest distance to the user with lowest price. In this example, the attributes like price and coordinates of the place is easy to be added into database in advance, but adding the attribute of user's interest of Qing dynasty as a value in database is not easy. Usually, each user has so different preference that no one database can include every attribute of user's preference. Obviously, if user's preference in not included completely as attributes of a place in database, the place that may be the candidate interesting place should not be recommended to the user.

Another problem is, in most cases, mobile user usually does not provide all attributes of his preference or query. In extreme case, especially when mobile user is moving, he even would not like to provide any preference or query explicitly, or register himself in a service system. For example, when a user goes a city on business at first time without any schedule for tourism, he will even not know where he wants to visit and how to provide an effective preference or query to get those interesting places for him. How does it work for information recommendation system without user's preference? If the system can automatically deal with the user's implicit preference without user's explicitly providing preference or query, then recommend effectively some interesting places especial in the typical case of when user is moving.

The problems that mobile information recommendation system meets mentioned above motivates us to design an effective location based mobile information recommendation system for mobile users, providing the most possible interesting places to the user when he is moving, according to his implicit preference and physical moving location without the user's providing his preference or query explicitly.

First, for the attributes that can not be included in database, we keep it as a form of text. Borrowing the idea of mobile information retrieval that takes account of the location and behavior of the user to improve the geographic relevance [7] of the information retrieved by a mobile user, we assume that the place should own three kinds of characters: (1) geographic feature like coordinates, (2) simple attribute, like price, (3) related document, that means the text which describes the information about the place and could not be included as a value into database. The document will be preprocessed at semantic level in order to match the user's interest. Section 4 will give its theoretic explanation.

According to the assumption, we propose two circle concepts. One is a physical position circle that represents spatial area around the user, formed by characters (1) and (2) if using the skyline query to calculate. The other is a virtual preference circle that is a non-spatial area related to user's preference of interests. A skyline query place in physical position circle, if its semantic meaning of document matches mobile user's implicit preference in virtual preference circle as well, the place will be recommended as a candidate of user's interest. The two kinds of circles refer to the Fig. 1.

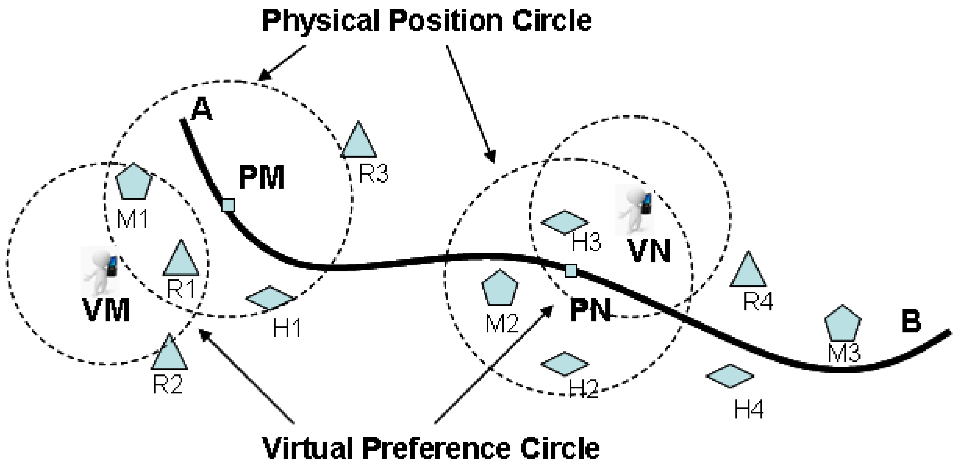


Fig. 1: Scenario of location based information recommendation according to two circles

In order to implement a real-time recommendation, system does some pre-processes offline before the virtual preference circle calculation. The pre-process includes user's implicit preference estimation that automatically done by system as a series of important terms according to user's historical visiting behaviors [8], and documents similarity matrix that reveals the relationship of similarity among all documents which are responsible to describe the places. To save the time of documents similarity matrix calculation, we just calculate the matrix for those documents that are most similar. We use semantic clustering method [9] to cluster all documents of places, only selecting the cluster which is most similar to the user's preference to calculate the matrix. Research results showed pseudo-relevance feedback is very popular and used by almost all high-performing research systems [10, 11]. If new features are added to the initial query, the revised query will search more relevant results on average when the revised query re-runs. Automatic query expansion requiring two rounds of retrieval and document analysis makes it runs typically vastly 20 or more times slower for some systems [10]. This slowdown prevents the adoption of automatic query expansion for real information recommendation systems. The similarity matrix will be used for automatic query expansion [10] in order to get the real-time response time and retrieval results in mobile environment.

Such preprocess benefits the real-time process to form the virtual preference circle. We use the implicitly estimated user's query or explicit query if user provides to match content of each document in cluster, the degree of similarity is calculated to determine whether the documents from the most similar cluster is used to expand the estimated user's query. Because the document similarity matrix has been already existed, the process of query expansion is so fast that satisfies the real-time information process requirement [10]. The top ranked documents returned by information retrieval method are put into the virtual preference circle, marked by a number of each place to explain which place the document belongs to.

Fig. 1 gives a promising scenario of location-based information recommendation which tends to recommend appropriate places based on the user's situation and preferences to the user even without his explicit query, and shows the two circles in it. Now let us suppose that as a mobile user is moving from place A to place B, the bold line represents the user's trajectory in Fig. 1. When he moves from A to current location PM where is decided by GPS signal, suppose that most users like the lowest price of a place, like entrance fee of a museum or dinner price. According to the attributes of place's coordinates and providing lowest price, our recommendation system uses skyline query to calculate the places in a physical position circle, with the center of user current location PM and a radius of a certain distance the user can arrive in a short time, for example, 1 km if user goes by foot. From Fig. 1, at the current location PM, the places M1, R1, H1 are in the physical position circle. M, R, H represents museum, restaurant, and hotel respectively. The virtual preference circle can be formed using the method mentioned above, which consider the user's preference as circle center VM. We observe that the documents of places M1, R1 are in this circle to illustrate the two places match the user's preference better than those of other places. We can recommend those places M1, R1 in the overlapping area between the two circles to mobile user when he is located on the place PM. When the user goes on moving from PM to PN, his virtual preference circle center also moves to VN, in this case, the place H3 in the overlapping area is recommended to the user. Based on this scenario, we define system the Location-based Recommendation System based on Skyline Query and Information Retrieval (LRS-SQIR)".

This paper aims to make contributions on the following aspects:

(1) User's preference or query automatic generation: mobile user is not compulsorily required to provide his preference directly and explicitly. We propose a method that is capable of learning user preferences in order to reduce the burden of user's input. Even in the case of without user's providing directly preference or query, our system can form preference query for mobile users.

(2) Integrate effective language model of information retrieval into recommendation system: We integrate automatic query expansion based on relevance model so that we find the virtual preference circle more accurately and effectively in a real-time environment. The offline calculation of document similarity matrix reduces the time needed for automatic query expansion, it provides almost no delay at recommendation time.

(3) We only use coordinates and price as default attributes to save the time of the calculation of skyline query. As to other attributes that are not easy to be quantitated as a value, we use information retrieval method to rank the documents of each place. According to the flexible attributes match besides the geographic coordinates, our system has ability to recommend different type of places, like museum and hotel for mobile user when he is moving, as the normal skyline query just recommends one type of place, such as hotel only, for user at a calculation time.

The rest of this paper is organized as follows: In Section 2 we give some related works on mobile information recommendation and information retrieval. In Section 3, we give an architecture of our LRS-SQIR system. Section 4 is the methodologies used in this paper. Section 5 is the experiments and analysis. In Section 6, we conclude and present future work.

II. Related Work

Many skyline query algorithms have been proposed for different types of databases and different situations. Skyline query is also used for information recommendation system.

Kodama et al. [3] also proposed a location based approach to recommending restaurants to a mobile user, by taking into account his current location and preferences. This approach showed a good recommendation effects, but there is still at least two problems in it. One problem is the skyline may contain a small number of objects when one or two objects are very strong compared to other objects, or when the number of attributes is small [3]. Though they proposed the second algorithm to solve this problem, it still needs the interactivity with user and requires more time to calculate the skyline query again. Another problem is it dealt with one type of object, like restaurant, in a computing time, furthermore, if some attributes of the object are not in database, the approach can not find out the object. We improve their approach by taking into account the information retrieval in recommendation system. That is, we only take coordinates and price to fast calculate the skyline query. We will retrieve other attributes that may be not easy to input the relational database by user query expansion. According to retrieve the attributes similar to user's preference by information retrieval method, and reduce the constraints of skyline query attributes, different type of places can be provided by our system.

Kossmann et al.[12] presented an online algorithm to returns the first skyline results immediately, allowing the user to give preferences during the running time of the algorithm so that the user can control what kind of results are produced next. Their work focused on the guide for skyline query from user interaction. The difference is our work does not require user interaction, we consider it is not convenient for mobile user to interact with system especially when he is moving. This is why we tend to implicitly estimate user's preference by using user's visiting history. Through preprocessing work, our system can provide real-time recommendation with a simple skyline query which only takes into account the attributes of coordinates and price of a place.

Using independent component analysis (ICA) [13], a document can be viewed as being generated based on an interaction of a set of independent hidden topics, thus an ICA model in text data analysis can be used to separate these independent hidden topics. With this understanding, Kolenda [14] used ICA to perform documents clustering in the latent semantic space according to each latent topic with an open source toolkit. Pu et al. [9] employed ICA to cluster documents in latent semantic space, and estimated a relevance model using documents in activated semantic cluster. Zhukov et al. [15] investigated applications of ICA and PCA for soft clustering and topic identification. PCA is good at reducing the dimensionality of the data, while ICA provides superior identification of the topics.

Lavrenko et al. [10] showed that by expending time for calculating a set of affinity lists – cross-entropy $H(M \|D)$, for every document M in the collection during indexing, the time needed for automatic query expansion can be reduced to the point where it provides almost no delay at query time. The key point is that it's not necessary to know the affinity between every pair of documents in the collection, they found that it is sufficient compute $H(M \|D)$ for most similar documents and expect that the query Q_M will bring those documents to the top of the ranked list. The initial query Q_M is formed by representative words from document M. After using Q_M to retrieve a set of top-ranked documents, the affinity $H(M \|D)$ is computed and stored as a documents similarity matrix. Our method borrowed this idea of offline computing, the difference is the method of selecting documents to compute the affinity. We select the documents from a semantic cluster to calculate the documents similarity matrix, because we believe that documents from the semantic cluster are more similar to user query than those documents coming from a pseudo query by representative words of a document.

III. Architecture of LRS-SQIR

The architecture of LRS-SQIR we design contains three main modules: preprocessing module, query processing module and visual module, see Fig. 2. There are two kind of database in this system, one database is the traditional relational database in which the coordinates of place and its price are stored. The other is the document of place corpus which has been indexed and some related documents similarity matrix has also been calculated.

All system is divided into two kinds of calculations: online and offline calculation. The offline calculation is mainly managed by the preprocessing module.

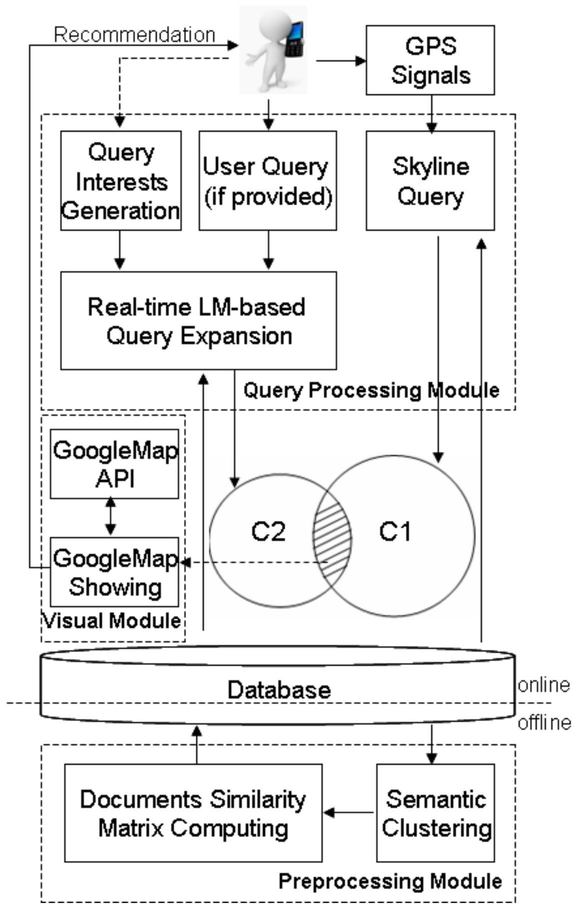


Fig. 2: Architecture of LRS-SQIR system

The outline of workflow in Fig. 2 is described like this: for the offline part, preprocessing module clusters documents in corpus using a semantic clustering method, then selects documents with high similarity score in a cluster that is similar to user's preference to compute documents similarity matrix. The matrix and all documents index will be stored in database.

When a mobile user is using this system, the system will online obtain his location coordinates by using GPS signals. This coordinates will be put into skyline query with lowest price and the coordinates from database, given a certain Euclidean radius distance, the result of places are in the physical position circle, denoted as C1. Meanwhile, the system estimates user's preference or query according to his visiting history records, this step can be also done offline. If mobile user provides his query, the user explicit query combing with the implicit user's preference will be expanded in a real-time by using language model based method. The real-time process is guaranteed by the documents similarity matrix existed in database in advance. The result of documents which belongs to places is in the virtual preference circle, denoted as C2. The visual module is responsible for recommending and visualizing the places in the shadowy area by calling Google map API.

IV. Methodology

Generally we give the algorithm of how to recommend information if the mobile user does not provide his preference or query explicitly, see Algorithm 1.

In Algorithm 1, we only input the system the documents collection of places, the database that stores the coordinates and price of places, and user visiting history. The function form_query(u, h) generates automatically user preference or query q according to user's visiting history; the function s_clustering(d) uses independent component analysis based semantic clustering method to clustering the documents of places in collection into C clusters; the function dsm(C,q) computes the documents similarity matrix M; the function rqe(q, M) ranks the documents of places by the similarity score with comparing to user's preference model, using user's query expansion method, the ranking results form a virtual reference circle *VRC*; the function skyline_query(l, p) will get the places that is nearest to user and provides the lowest price, the results of skyline query form a physical position circle *PPC*. If the place or its document is in the overlapping area of the two circles at the same time, it will be recommended virtually to the user by calling GoogleAPI.

According to the generic algorithm, we give the methodologies used in this paper as follows:

4.1 Estimation of query user preference or query model

User implicit query may be represented by user's visiting history behaviors, such as a list of terms and weights associated with those terms, a list of visited URLs and a list of past search queries and pages clicked for these search queries. We look each behavior as a document with a certain preference degree. We define mobile user's query as function of interesting documents and corresponding preference degree like [16]:

$$UserQuery = f\left(<D_1, g_1>,\ \dots\ ,<D_i, g_i>\right) \tag{1}$$

where $<D_i, g_i>$ denotes what preference degree g_i is used by user to visit document D_i. The preference degree is defined like: $g_i = g(D_i, Act_i, U)$, $0 \le g_i \le 1$, U represents user. $Act_i = (a_1, a_2, \dots, a_n)$, $i = 1, 2, \dots, n$, is a user visiting activity vector, such as register information, browse-pages, browse-time, query terms, download pages, bookmarks and etc, each Act_i has different interest level a_i according to empirical data, $0< a_i \le 1$.

Algorithm 1 Process of LRS-SQIR

```
1:   Function LRSQIR(u, h, d, db)
2:                                         //u: mobile user
3:                               //h: user's visiting history
4:                       //d: documents of places in collection
5:                  //db: store coordinates and price of places
6:     q ← form_query(u, h)
7:                              //form query for user u by his h
8:     C ← s_clustering(d)
9:                                   //semantic clustering for d
10:    M ← dsm(C, q)
11:                          //get documents similarity matrix M
12:    VRC ← rqe(q, M)
13:                            //get virtual reference circle VRC
14:    PPC ← skyline_query(l, p)
15:                                  //l: coordinates of place
16:                                          //p: price of place
17:                           //get physical position circle PPC
18:    for each place in VRC ∩ PPC
19:      GoogleAPI(place,u)
20:                        //visually recommend place to user
21:    end for
22:    end function
```

We calculate expectation of all activities: $E(\boldsymbol{Act}_i) = \sum p_i a_i$, p_i is the weight of activity i, $\sum_{i=1}^{n} p_i = 1$. Finally, the user query model is denoted as a vector space model [17] as below:

$$\begin{aligned} UserQuery &= \sum\nolimits_{i=1}^{n} g_i D_i \\ &= \left(\sum\nolimits_{i=1}^{n} g_i w_{i1}, \sum\nolimits_{i=1}^{n} g_i w_{i2}, \cdots, \sum\nolimits_{i=1}^{n} g_i w_{im}\right) \\ &= \left(\sum\nolimits_{i=1}^{n} p_i a_i w_{i1}, \sum\nolimits_{i=1}^{n} p_i a_i w_{i2}, \cdots, \sum\nolimits_{i=1}^{n} p_i a_i w_{im}\right) \\ &= \left(u_1, u_2, \cdots, u_m\right) \end{aligned} \tag{2}$$

where w_{im} represents weight of term m in document i, which is calculated by tf-idf formula [17].

$$w_{im} = \frac{tf_{im} \log\left(N/n_m + 0.01\right)}{\sqrt{\sum_{m=1}^{n} \left(tf_{im}\right)^2 \left[\log\left(N/n_m + 0.01\right)\right]^2}} \tag{3}$$

where tf_{im} is term frequency of term m in document i, N is the number of total documents, and n_m is the number of document containing term m.

4.2 Independent component analysis based semantic clustering

For effectively computing the documents similarity matrix, like in [10], we do not need to calculate the every pair of documents in the collection. We are going to only computing the pair of documents that are most similar to user's implicit preference. In order to find these documents, we use independent component analysis (ICA) [13] based semantic clustering method to clustering the documents of places in collection. ICA is a suitable approach to semantic clustering for a set of documents [9, 13, 14], The ICA model in text data analysis can be described as follows:

$$\boldsymbol{X} = \boldsymbol{AS} \tag{4}$$

where $\boldsymbol{X}$ is a term-document matrix that holds m observed terms in each row with n document samples in each column, $\boldsymbol{X}=\{x_1, x_2, \ldots, x_n\}$, m is the same size of the vocabulary. $\boldsymbol{A}$ is an unknown $m\times k$ mixing matrix with non-orthogonal transformation basis, and $\boldsymbol{S}$ is another unknown matrix that holds k independent components (latent topics) in each row and n document samples in each column. If we can derive the inverse matrix $\boldsymbol{A}^{-1}$, then $\boldsymbol{S} = \boldsymbol{A}^{-1}\boldsymbol{X}$, usually denote the matrix $\boldsymbol{A}^{-1}$ as an unmixing matrix $\boldsymbol{W}$, that is, $\boldsymbol{W}=\boldsymbol{A}^{-1}$.

$$\begin{array}{ccccc} & \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_n \\ C_1 & p_{11} & p_{12} & \cdots & p_{1n} \\ C_2 & p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ C_k & p_{k1} & p_{k2} & \cdots & p_{kn} \end{array}$$

Fig. 3: Semantic clustering matrix

According to the matrix $\boldsymbol{S}$, we can evolve a $k\times n$ partition matrix $\boldsymbol{U}(\boldsymbol{X})$ of the term-document matrix $\boldsymbol{X}$ in $\Re^N$. The matrix $\boldsymbol{U}(\boldsymbol{X})$ represents the partitioning into a number k of semantic clusters ($C_1, C_2, \ldots, C_k$), it can be denoted as $\boldsymbol{U}=[u_{ij}]$, i=1, …, k, and j=1, …, n, where u_{ij} is the membership of document x_j belonging to a semantic cluster C_i. Fig. 3 shows that the element p_{ij} in semantic clustering matrix $\boldsymbol{U}(\boldsymbol{X})$ represents document x_j belongs to semantic cluster C_i with a probability p_{ij}. With the help of such a probability, a document $\boldsymbol{x}_j$ could be assigned to a latent cluster C_i according to the highest probability p_{ij}, as shown formally as

$$p\left(\boldsymbol{x}_j \mid C_i\right) = \arg\max_i p_{ij} \tag{5}$$

According to equation (5), the clustering of documents therefore boils down to a cluster label assignment to each document based on the obtained maximum probability in Fig. 3 matrix.

There are k clusters after semantic clustering on the documents collection. We only select the documents in cluster that is most similar to the user's implicit preference to compute the documents similarity matrix. Formula (6) is a Kullback-Leibler (KL) divergence that measures the closeness between the two models and gives how to select this cluster by match the similarity between user's preference and cluster.

$$\arg\min_{\theta_S} D\left(\theta_Q \square \theta_S\right) = \sum_{\substack{w\in V \\ \theta_S \in \Theta_S}} p\left(w \mid \theta_Q\right) \log \frac{p\left(w \mid \theta_Q\right)}{p\left(w \mid \theta_S\right)} \tag{6}$$

where θ_Q is the a user's preference or query model, θ_S is the semantic cluster model which is estimated from a semantic cluster. Θ_S represents all the semantic clusters derived from ICA algorithm. The semantic cluster that has the smallest KL divergence value to user's query would be used for computing document similarity matrix. The value of every pair of documents in the selected cluster is represented as the cross-entropy of their document language model [10], shown as follows.

$$H\left(\theta_{d_i} \,\|\, \theta_{d_j}\right) = \sum_w p\left(w \mid \theta_{d_i}\right) \log p\left(w \mid \theta_{d_j}\right) \tag{7}$$

where θ_{d_i}, θ_{d_j} are document model of document d_i and d_j.

4.3 Retrieval with user query expansion

Relevance models [18] compute the probability that the word would appear during random sampling from same distribution that produced the query. Though automatic query expansion can improve the retrieval results, it is slow [10] that prevents it from adoption for real retrieval systems, especially in mobile environment. According to [10], good retrieval performance can be achieved if we rank the documents D in the collection by the cross-entropy of their language model θ_D with the estimated relevance model θ_R:

$$H\left(\theta_R \,\|\, \theta_D\right) = \sum_w p\left(w \mid \theta_R\right) \log p\left(w \mid \theta_D\right) \tag{8}$$

Because the summation goes over all vocabulary words w in equation (8), running a query consisting of thousands of words is very inefficient and requires a lot of computational resources [10]. If directly using equation (8) in our information recommendation system, it will meet a real-time problem. After transformation of the equation (8), we get:

$$\begin{aligned} & H\left(\theta_R \,\|\, \theta_D\right) \\ & = \sum_w \log p\left(w \mid \theta_D\right) \left(\sum_{\theta_M} p\left(w \mid \theta_M\right) p\left(\theta_M \mid q_1 \ldots q_k\right) \right) \\ & = \sum_{\theta_M} H\left(\theta_M \,\|\, \theta_D\right) p\left(\theta_M \mid q_1 \ldots q_k\right) \end{aligned} \tag{9}$$

Notice that $H(\theta_M \| \theta_D)$ in equation (9) is independent of the user's query and can be pre-computed and stored at the time when the collection is indexed [10]. We give the algorithm for computing the documents similarity matrix offline. We do only calculation the $H(\theta_M \| \theta_D)$ for the pair of documents which is most similar to user's preference. Algorithm 2 gives the effective process of document similarity matrix calculation.

Algorithm 2 Document Similarity Matrix

```
1:  function DSM($d$, $q$)
2:                                  //$d$: each document in collection
3:                                  //$q$: user's preference or query
4:    $M \leftarrow \emptyset$             //document similarity matrix
5:    $C \leftarrow$ ica($d$)               //semantic clustering
6:    $max \leftarrow 0$          //maximum score of similarity
7:    for each $c$ in $C$
8:      $score \leftarrow$ sim($c$, $q$)
9:                                  //similarity between $c$ and $q$
10:     if $score > max$ then
11:       $max \leftarrow score$
12:     end if
13:   end for
14:   $c \leftarrow$ get_cluster($max$)
15:                                 //get most similar cluster, $c \in C$
16:   for each $d_i$, $d_j$ in $c$
17:     $m_{ij} \leftarrow H(d_i \| d_j)$
18:                                 //cross-entropy calculation, $m_{ij} \in M$
19:   end for
20:   return $M$
21: end function
```

In Algorithm 2, the function DSM(d,q) computes the document similarity matrix. M represents the resulting document similarity matrix. The function ica(d) makes semantic clustering for all documents in collection, the semantic clustering method refers to [9, 14]. The function sim(c,q) computes the similarity between semantic cluster c and user's query q using the formula (6). We select the most similar cluster by function get_cluster(max) to calculate the cross-entropy $H(d_i \| d_j)$ between documents in the cluster. Finally, we store the result of document similarity matrix M.

According to [10], we give the Algorithm 3 for effective query expansion for refining user's preference and rank documents of places according to its similarity to the user's information need.

Algorithm 3 Ranking With Query Expansion

```
1:  function RQE($Q$, $M$)
2:                                  //$Q$: user's preference or query
3:                                  //$M$: document similarity matrix
4:    $RankList \leftarrow \emptyset$
5:                                  //array to store ranked documents
6:    $TopK \leftarrow$ topk($Q$)
7:                                  //get top-ranked $K$ documents by $Q$
8:    $Sum \leftarrow \emptyset$
9:                                  //array to store summation of cross-entropy
10:   for each $d_i$ in $M$
11:     $Sum[i] \leftarrow 0$        //initialize summation for $d_i$
12:     for each $d_j$ in $TopK$              //$1 \le j \le k$
13:       $Sum[i]$ += $m_{ij} * p(d_j \mid Q)$
14:                                 //ranking score of $d_i$
15:     end for
16:   end for
17:   $RankList \leftarrow$ sort($Sum$)   //sort by ranking score
18:   return $RankList$
19: end function
```

In Algorithm 3, the function RQE(Q,M) ranks the documents of places with considering user query expansion. Q is user's preference or query, $Q = (q_1,q_2,\ldots q_k)$, M represents the resulting document similarity matrix. The function top(Q) uses basic language model [19] to retrieve top-ranked K documents from collection by user's query Q. Line 13 is the implementation of Formula (9). The ranking score of document d_i is stored in $Sum[i]$. The calculation method of $p(d_j \mid Q)$ refers to [18]. The final rank list is sorted according ranking scores in descending order.

4.4 Skyline query

A skyline query is a query to select the set S of all the objects such that are not dominated by other objects [2]. S is called the skyline and formally given as follows:

$$S(O) = \{o \in O \mid \neg\exists o' \in (O - \{o\}) \wedge o' < \mathrm{o}\} \qquad (10)$$

The '<' in equation (10) means dominance relationship [3]. Given two objects O_1 and O_2, if O_1 is equal to or better than O_2 in terms of all attributes, and if O_1 is better than O_2 at least one attribute, we say O_1 dominates O_2, and write $O_1 < O_2$.

In our location based recommendation system, we always want the places in order from the nearest to further places. If a lot of attributes are added to calculate the skyline, it is a time-consuming task. We only take the attributes of price into account to determine the dominance relationship.

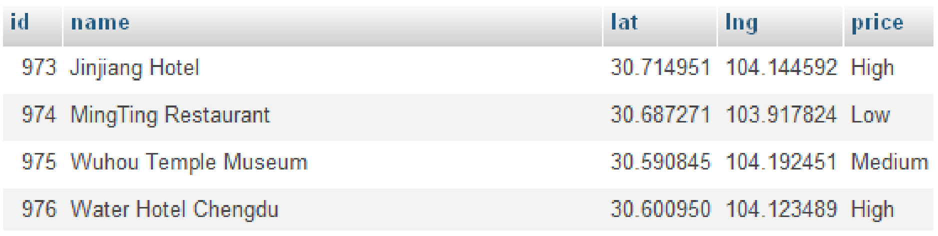
```
SELECT *
FROM Etourism
WHERE city = 'Chengdu'
SKYLINE OF price MIN, distance MIN;
```

Fig. 4: Example of skyline query

We can use the skyline query [2] as Fig. 4 to form the physical position circle in which the places are nearest to the user and provide the lowest price.

V. Experimental Results and Analysis

To examine the effects of our location based information recommendation system, we designed the following experiments. We prepared the data for simulating the environment at the first step. For the distribution of different places in a city, like in Chengdu, China, we randomly generated coordinates in Great Chengdu city area for 1000 places that represent hotels, restaurants, museums and so on, the same dealing with price of places was also done. We generated the virtual information because our target is to test if our method works better or not, it is not important for places if we put a virtually generated on it. All the data were store in MySql database as Fig. 5 shows. When a skyline query like Fig. 4 is acted on the data in Fig. 5, as a result, a physical position circle will be formed in which the places are nearest to user as well as has lowest price.



id	name	lat	lng	price
973	Jinjiang Hotel	30.714951	104.144592	High
974	MingTing Restaurant	30.687271	103.917824	Low
975	Wuhou Temple Museum	30.590845	104.192451	Medium
976	Water Hotel Chengdu	30.600950	104.123489	High

Fig. 5: Random generation of places in Chengdu city

For forming virtual preference circle, we conducted experiments on one TREC data set: Los Angeles Times (LA) with query topics 301-400 except query 320 and query 334 which have no relevant documents in the judged pool for the LA collection. Indri 2.9 system was used for indexing and retrieval. The collection and all 98 queries were stemmed using Porter stemmer, and stop-words were removed as well.

Usually, many experiments need to verify the recommendation of place is right or not manually through a group of volunteers, according to their preference judgment. Suppose that we have documents of places, we can organize them into a collection and do the indexing and retrieval. To save the time and avoid some judgment errors happening due to the boring evaluation process, we decide to automatically judge if the document of a place meets the user's preference or not. So we used the relevant document in judged pool of LA collection as the documents that describe the attributes of places. For each place, we only assigned relevant documents of one query to a place, and marked the documents with the *id* of place. For example, we assigned documents $d_1 \ldots d_{20}$ of query 310 with *id* 974, that means, the 20 documents are the documents of place "MingTing Restaurant", see Fig. 5.

These relevant documents assigning to each place are unknown by users, if user's preference or query can retrieve some of the relevant documents, we think that the place connecting to the retrieved documents should be in the virtual preference circle.

To effectively use the relevant documents as standard of judgment, we used query topics 301-400 for LA collection as simulation of the user's preference or query after estimation according to user visiting history.

The implementation of ICA algorithm for documents semantic clustering was from *DTU*: *Toolbox* [20]. After then, the documents similarity matrix was calculated and stored. The documents are from a semantic cluster that is most similar to the user's query.

The initial user query results were generated using the basic query likelihood language model. Top 50 documents were used to do a query expansion. Table 1 gives the time comparisons of query running and documents similarity matrix computing as well as the average precision of retrieval by using queries 301-400 on collection LA among the basic language model (LM) [19], the relevance model with full initial retrieved results (LM-full) [18]as feedback for query expansion, the fast relevance model (fRM) [10], the semantic relevance model (SRM) [9] and the fast semantic relevance model (fSRM). The data in columns LM, RM-full, fRm are directly from [10] for comparison.

From Table 1, we observe LM method is the fastest one that only takes 7.6 seconds to fulfill 100 queries. fSRM method takes 11.75 seconds vs. fRM method's 15.83 seconds for completing the whole 100 queries. For fSRM method, a query only needs to 1/10 of a second to finish. The fast speed shows the three methods are suitable for real-time retrieval task, like mobile information recommendation. Our fSRM method outperforms the other two, because it takes acceptable time to deal with the user's queries and retrieves more precise results. It gets average precision 27.34% that is better than the RM-full method 27.19% and the LM method 24.58%. This comparison presents that our fSRM retrieving information in a real time is a good choice for mobile information recommendation that has ability to provide almost no delay at recommendation time.

Though RM-full method and SRM get higher average precision, their retrieving time is also long, achieving 611.43 seconds and 829.53 seconds. The long retrieval time is caused by the query expansion and document content process. SRM needs more time because it requires another step for clustering the documents from top *N* initial retrieved results.

Table 1: Time comparison of query running and documents similarity matrix computing on collection LA

	LM	RM-full	fRM	SRM	fSRM
Ave Pr	0.2458	0.2719	0.2653	0.2958	0.2734
time[s]	7.6	611.43	15.83	829.53	11.75
sims[h]			38.55		13.47

The LA collection required over 38 hours of time to build the document similarity matrix if using fRM method. If a collection will be indexed rarely and searched often, this is a fine tradeoff. In other cases, the indexing time may be unacceptable [10]. For our mobile recommendation system, we would like to use the real-time query expansion in information retrieval area as well as need less time to create the document similarity matrix. Our fSRM method shows that it only needed almost 1/3 of fRM building time for the matrix. We analyze the reason of the amount of time reduction is we use fewer documents for calculation the documents similarity matrix. fRM used all the top *N* documents while we just used some of top *N* documents in a semantic cluster to calculate. One semantic cluster has at most *N* documents, normally, it has average number of *N*/3 documents in the semantic cluster that is most similar to user's preference or query. As we just take into account the most similar documents to calculate, it's not strange using our fSRM method to building the matrix during indexing time will be cut to a large degree when comparing to fRM method.

D	Skyline points
2	60
3	783
4	4515
5	13022
6	20000

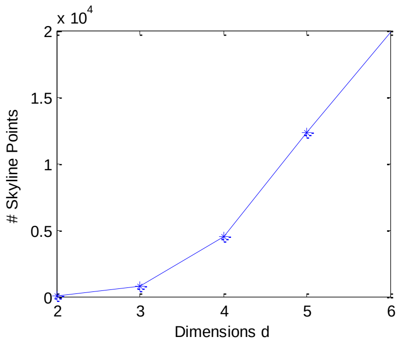


Fig. 6: Skyline size

The mobile information recommendation system requires a real-time response to user's query. This is the reason why we only took into account two attributes of a place, coordinates and price, to calculate the skyline query, forming the physical position cycle. We examined the skyline size and skyline query calculation time to prove it is reasonable that we consider more than 2 attributes using high effective information retrieval method. We also extended our dataset from 1000 places to 20000 places as our test bed. The attributes are also increased from 2 to 6.

Fig. 6 shows the skyline size. When taking two attributes or dimensions of a place into computing, we got 60 skyline points there. As attributes increased to 3, there are totally 783 skyline points, 11 times than that of using dimension 2. As the attributes achieved to 6, all the places in dataset became the skyline points. An increasing number of skyline points are involved in recommendation system, the user faces a second or third round to decide what place is his really interesting place.

d	Times [s]
2	2
3	37
4	94
5	203

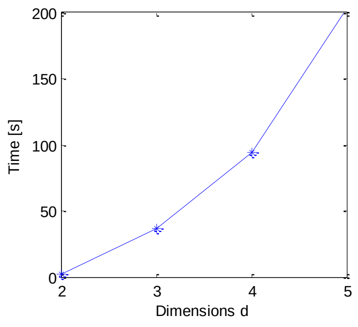


Fig. 7: Running time using BNL-basic algorithm

Fig. 7 gives the skyline query running time using the BNL-basic algorithm [2]. We observed the running time will increase explosively as the dimension increases. If we choose 4 attributes to compute the skyline query, it will cost us 94 seconds to get more than 4500 skyline points. Considering the response time, it is obviously not good for mobile information recommendation system. If using our method in this paper, the effective information retrieval method saved a large amount of time to deal with the attributes that will take too much time and bring a lot of skyline results in skyline query. According to the experiments, our mobile information recommendation system can response a user's query in 2 seconds.

VI. Conclusion

In this paper, we presented a mobile information recommendation system to provide interesting places, such as restaurants, hotels, museums and so on to mobile users without their explicitly providing his preference or query. Our method is capable of learning user preferences and forming user query according to his visiting history. We integrated automatic query expansion based on relevance model so that we can find the virtual preference circle more accurately and effectively in a real-time environment. The offline calculation of document similarity matrix according to semantic cluster reduced the time needed for automatic query expansion it provides almost no delay at recommendation time. With the help of information retrieval method to make use of the attributes that can not be added in relational database, we used a simple skyline query with only two attributes of coordinates and price to save the time of the calculation of skyline query. Our main contribution is to combine skyline query and information retrieval method together to form two kinds of circles that a user physically or virtually located in. This combination makes our location based information recommendation system work better.

Acknowledgments

This work was partially supported by European Commission in Sustainable e-Tourism: Erasmus

Mundus Action 2 Program, Science & Technology Foundation of Chengdu University under Grant No. 2011XJZ01, Science & Technology Project of Sichuan, China under Grant No. 2010SZ0004 and NSF under NSF/IIS 0704628.

References

[1] Lbath, A. Method and Device for Automatic Production of Context Aware Mobile Services. Pat. N° WO 006721, 2005. http://patentscope.wipo.int/search/en/WO2007006721.

[2] Börzsönyi, S., Kossmann, D., and Stocker, K. The skyline operator. in Proceedings of the 17th International Conference on Data Engineering. 2001. Heidelberg, Germany.

[3] Kodama, K., et al. Skyline Queries Based on User Locations and Preferences for Making Location-Based Recommendations. in ACM LBSN'09. 2009. Seattle, WA, USA.

[4] Boulmakoul, A., L. Karim, and A. Lbath, Moving Object Trajectories Meta-Model And Spatio-Temporal Queries. International Journal of Database Management Systems (IJDMS). 2012. 4(2)

[5] Huang, Z., et al. Skyline queries against mobile lightweight devices in MANETs. in ICDE'06. 2006.

[6] Huang, X. and C.S. Jensen. In-Route Skyline Querying for Location-Based Services. in LNCS 3428. 2005.

[7] Raper, J., Geographic relevance. Journal of Documentation, 2007. 63(6): p. 836-852.

[8] Kuo, M.-H., Chen, L.-C. , and Liang, C.-W. Building and Evaluating a Location-Based Service Recommendation System with a Preference Adjustment Mechanism. Expert Systems with Applications, 2009. 36: p. 3543–3554.

[9] Pu, Q. and He, D. Semantic clustering based relevance language model. Information Technology Journal, 2009. 9(2): p. 236-246.

[10] Lavrenko, V. and Allan, J. Realtime query expansion in relevance models. in IR 473. 2006. University of Massachusetts.

[11] Kurland, O., Lee, L., and Domshlak, C. Better than the real thing? Iterative pseudo-query processing using cluster-based language models, in Proc. 28th ACM SIGIR conference on Research and Development in Information Retrieval. 2005: August 15-19. p. 19-26.

[12] Kossmann, D., Ramsak, F., and Rost, S. Shooting Stars in the Sky: An Online Algorithm for Skyline Queries. in Proceedings of the 28th VLDB Conference. 2002. HongKong, China.

[13] Hyvärinen, A., Survey on independent component analysis, in Neural Computing Surveys 2. 1999. p. 94-128.

[14] Kolenda, T., Clustering text using independent component analysis. 2002, Inst. of Informatics and Mathematical Modeling, Tech. University of Denmark.

[15] Zhukov, L. and Gleich, D. Topic indentification in soft clustering using PCA and ICA, in Technical report. 2004, Yahoo Research Labs.

[16] Pu, Q., et al., Information recommendation using improved feature selection and independent component analysis. Journal of Computational Information Systems, 2007. Vol. 3(4): p. 1731-1738.

[17] Salton, G., and Buckley, B. Term-weighting approaches in automatic text retrieval. Information Processing and Management, 1988. 24: p. 513-523.

[18] Lavrenko, V., and Croft, W.B. Relevance-based language models, in Proc. 24th ACM SIGIR conference on Research and Development in Information Retrieval. 2001: New Orleans, LA, USA. p. 120-127.

[19] Ponte, J., and Croft, W.B. A language modeling approach to information retrieval. in Proc. 21st ACM SIGIR conference on Research and Development in Information Retrieval. 1998. Melbourne, Australia.

[20] Kolenda, T., et al., Dtu:toolbox, in http://isp.imm.dtu.dk /toolbox/. 2002: Internet site, Informatics and Mathematical Modeling, Technical University of Denmark.

Qiang Pu: Associate Professor of School of Information Science and Technology in Chengdu University, main researcher in Key Lab of Pattern Recognition and Intelligent Information Processing in Chengdu University. He is now a Post-doctor in Joseph Fourier University, Grenoble 1, France. His current research focuses on pattern recognition, information retrieval, location based service and mobile tourism.

Ahmed Lbath: Professor of Joseph Fourier University of Grenoble, he is interested in GIS, location based service and mobile tourism.

Daqing He: Associate professor of school of information sciences in University of Pittsburgh, USA. His current research is cross-language information retrieval, machine translation.