

Locating Mapped Resources in Web 2.0

Dongxiang Zhang, Beng Chin Ooi, Anthony K. H. Tung

School of Computing, National University of Singapore

13 Computing Drive, 117417, Singapore

{zhangdo, ooibc, atung}@comp.nus.edu.sg

Abstract—Mapping mashups are emerging Web 2.0 applications in which data objects such as blogs, photos and videos from different sources are combined and marked in a map using APIs that are released by online mapping solutions such as Google and Yahoo Maps. These objects are typically associated with a set of tags capturing the embedded semantic and a set of coordinates indicating their geographical locations. Traditional web resource searching strategies are not effective in such an environment due to the lack of the gazetteer context in the tags. Instead, a better alternative approach is to locate an object by tag matching. However, the number of tags associated with each object is typically small, making it difficult for an object to capture the complete semantics in the query objects.

In this paper, we focus on the fundamental application of locating geographical resources and propose an efficient tag-centric query processing strategy. In particular, we aim to find a set of nearest co-located objects which together match the query tags. Given the fact that there could be large number of data objects and tags, we develop an efficient search algorithm that can scale up in terms of the number of objects and tags. Further, to ensure that the results are relevant, we also propose a geographical context sensitive *geo-tf-idf* ranking mechanism. Our experiments on synthetic data sets demonstrate its scalability while the experiments using the real life data set confirm its practicality.

I. INTRODUCTION

In Web 2.0, users are free to upload diverse kinds of resources and mark them in the map to indicate their relevance to the area using open map APIs. Such a large amount of user-contributed materials constitute a luxuriant spatial database that can provide immense mining opportunities. In this paper, we focus on the fundamental application of locating geographical resources.

The topic of detecting geographic locations [8], [17], [16], [1], [12], [14], [2] has been well studied in recent years. Effective location detecting technique has significant commercial potential and can assist the search engine in classifying and indexing the web resources to improve the relevance of the returned results. It also plays an important role in providing the customers with local personalized services. Existing work tackles this problem by mining and extracting phrases that contain geographical context in web documents or with the aid of hyperlink structures and query logs when the geographical context is not clear. The ambiguities on the location names are eliminated via NLP or IR technique to assign the correct scope so that the term such as “Washington” appearing in “Denzel Washington” will not be treated as a location name. Despite considerably high accuracy, traditional methods still face with new challenges in the Web 2.0 environment:

- Various types of resources exist in the spatial database. Existing search engines pay particular attention to gazetteer terms derived from web documents. However, other multimedia resources, such as photos and videos, are not associated with such terms inherently. Without the geographical context, traditional approaches may not work well.
- Current geographic information systems typically rely on the gazetteer information published by authorized communities. In Web 2.0, users have contributed huge amounts of useful contents collaboratively. A prominent example is Wikipedia, the most widely used online encyclopedia. There are abundant implicit geographic information embedded and they should be fully exploited.

In this paper, we propose to adopt tagging as a way to build a uniform data model for the mapped resources within the context of our Marcopolo system[6]. In Web 2.0, tagging is a popular means to annotate various resources, including news, blogs, speeches, photos and videos. Users are encouraged to add extra textual terms as semantic description or summarization for the objects. With human intelligence involved, the tags are well phrased so that much cost can be saved from handling term ambiguities. For example, “Denzel Washington” will be treated as a basic unit automatically without applying NLP or IR techniques. Although documents are essentially different from other media in textual context, tagging provides a means to build a uniform model to eliminate the difference:

Definition 1 (Uniform Mapped Resource Model): Let S be the d -dimensional geographical space and \mathcal{T} be the tag space. Each object o can be represented as $o = [ref, c_1, \dots, c_d, t_1, \dots, t_n]$ where $[c_1, \dots, c_d] \in S$, $t_i \in \mathcal{T}$ and ref is the reference to the object itself.

Based on this data model, the problem of locating mapped resources is essentially a spatial tag matching problem. Given a query object, we aim to find a spatial location that best matches the associated tags. Such a query also has great potential for location service providers. The service can be considered as a new type of mapped resource and represented in the form of tags. For example, fans of Apple’s products can submit “applestore subway” to locate a retailer store near the subway for convenient purchase of the products. Figure I illustrates the

spatial distribution of these two tags in New York City.¹ We can observe from the figure that Apple Store Fifth Avenue, which is located to the south-east of the Central Park is a location that gives a good match.

Finding co-locating tags in spatial databases remains an ongoing research problem. Traditional approaches of keyword search in spatial databases[10], [9], [7] are seeking for a mapped resource matching all the query tags. However, the number of tags associated with each object is typically small, making it difficult to find a complete match. On the other hand, these methods ignore the fact that spatially close resources could be belonging to the same object and are related to each other. For example, news about “New York City airplane river crash” could be marked by users around the crash location in the Hudson River and photos of “Statue of Liberty” are likely to be uploaded around the Liberty Island. Therefore, instead of looking for one-to-one match, we allow one query object to match multiple spatially correlated objects as long as the union of their tags can match all the query tags.

In our earlier work [18], the $mCK(m$ closest keywords) query is defined for finding a set of closest keywords in the spatial database. Since we are able to apply it in our uniform mapped resource model, we shall re-state the definition here:

Definition 2 (mCK Query Problem [18]): Given a d -dimensional spatial database $SD = \{o | o = [c_1, c_2, \dots, c_d, t]\}$ and a set of m query keywords $Q = \{t_{q_1}, t_{q_2}, \dots, t_{q_m}\}$, the mCK Query Problem is to find m tuples $\sqcup = \{o_1, o_2, \dots, o_m\}$, $o_i.t \in Q$ and $o_i.t \neq o_j.t$ if $i \neq j$, and $diam(\sqcup)$ is minimum.²

The closeness measure $diam(\sqcup)$ is defined as the maximum distance between any two tuples in \sqcup . The search algorithm in [18] shows good scalability in terms of the number of query keywords. However, the proposed bR^* -tree indexing structure requires the storage of auxiliary information for each possible tag and can therefore potentially incur high I/O cost when the tag space \mathcal{T} is large.

In this paper, we present a new and efficient index based on the R^* -tree[5] and inverted list. A labelled R^* -tree is constructed to provide spatial proximity information and the inverted list is used as a partition of the tag space \mathcal{T} . The augmented summary information is not stored but built dynamically during the search process to make the index lightweight. We design a bottom-up search algorithm to utilize the inverted index so that only the related lists will be accessed. Since the I/O cost has been greatly reduced, the new indexing and searching technique can ensure scalability in terms of both the number of query tags m and the data size within a large tag space \mathcal{T} .

In addition to the efficiency issue, we also address the issue

¹The data set is derived from Flickr’s geo-tagged photos and does not cover all the Apple retail stores and subways in NYC. In this paper, we assume a relatively complete geo-tag database has been built.

²A tuple o with multiple tags can be treated as multiple tuples at the same location: $o = \{o_i | o_i = [c_1, c_2, \dots, c_d, t_i]\}$

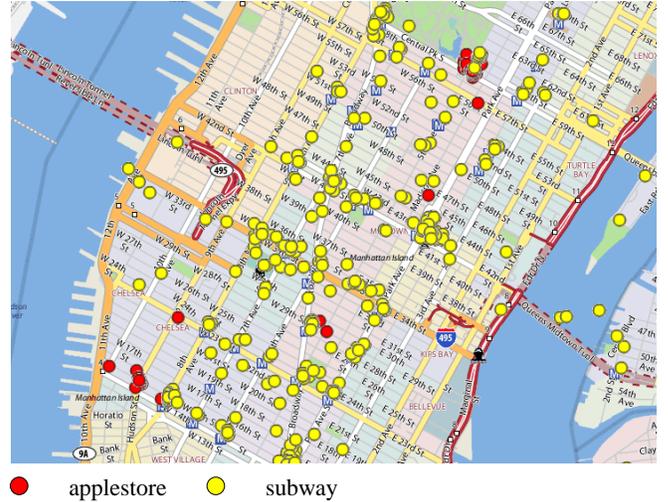


Fig. 1. Distribution of geo-tags “applestore, subway” in NYC

of semantic relevancy by proposing a re-ranking mechanism for co-located tags that are found within the top- k closest scope. To this end, we have to take into account the geographical context in the ranking process. There exist work in [8], [2] that propose geo-ranking mechanism using local popularity of web resources measured by citations. However, the hyperlink structure among the resources is not available in our data model. In the recent work of [7], Cong et al. propose to retrieve the most relevant spatial web objects by considering both the distance proximity and text relevance in the ranking function. However, the text relevance is still between query keywords and each single spatial web document. In this paper, we present a more general ranking method that takes the nearby resources into account as well. We extend the widely used $tf-idf$ and propose a new ranking strategy, named $geo-tf-idf$, to measure how the tags are related to the area that they are located in.

In summary, the main contributions of our paper include:

- We propose to use tags to build a general data model. Based on the model, we describe a system framework to support co-location searches on various types of resources in Web 2.0 applications.
- We develop an efficient indexing and searching strategy, which is scalable in terms of both the number of query tags and the data size of resources, to answer the queries of co-located tag matching.
- We extend the widely accepted $tf-idf$ method for the geographical context, called the $geo-tf-idf$ ranking method, to measure the relevancy of the geo-tags with respect to the area in which they are located.
- We conduct extensive experiments using synthetic and real life data sets. The results confirm the effectiveness and practicability of our proposal in the context of Web 2.0 applications.

The remaining of this paper is organized as follows. Section II discusses existing work on content based location searching and takes an overview of the work in answering mCK query .

Section III introduces the general framework and the improved index and search strategy. Section IV proposes the *geo-tf-idf* ranking mechanism. Section V presents our performance study on the synthetic and real life data sets. Finally, the paper is concluded in Section VI.

II. RELATED WORK

A. Content Based Location Searching

Location detecting service has attracted great interest and has been well studied [8], [17], [16], [1], [12], [14], [2] in recent years due to its commercial potential to the search engine in providing local or personalized service to customers. The works can be divided into two categories according to the source used to compute the geographical scope. The first category aims to extract and parse the addresses, location name, placemark, telephone and zip code from the web resource. Such gazetteer-based information extraction is obvious and effective. Natural language processing and information retrieval techniques have been applied for more effective retrieval of the gazetteer terms by eliminating the ambiguities so as to improve the recognition accuracy. On the other hand, when the geographical content is implicit, other sources, such as hyperlinks of web pages and log files, are utilized to detect the web page's location. A page that is popularly accessed or cited by other local pages or users is considered to be relevant to a local area. Unfortunately, in many Web 2.0 applications, these sources are not available in the database.

Recently, several works have been proposed to handle different types of keyword queries in spatial databases [10], [9], [7], [18]. Hariharan et al. [10] introduce a spatial keyword query with range constraints. Each spatial object returned is required to intersect with the query MBR (Minimum Bounding Rectangle) and match all the user-specified keywords. They propose a hybrid index of the R*-tree and inverted index, called the KR*-tree, to answer the query. Felipe et al. [9] propose a similar query type by combining *k*-NN query and keyword search, and use IR^2 , a hybrid index of the R-tree and signature file, for query processing. Cong et al. propose to take into account both location proximity and text relevancy during the ranking. They develop an efficient framework for top-k text retrieval. These work cannot be applied to search for co-located tags because they are looking for single objects matching all the user-specified tags. The query tags can appear in multiple tuples as long as these tuples are close and related in the geographical space. In [18], we address the problem of finding closest keywords in spatial databases. Its performance scales well in terms of the number of query keywords due to the proposed bR*-tree index and apriori-based search strategy. However, since it requires the storage of auxiliary information for each possible keyword in the indexing nodes, its index size can be enormous when a large number of tags are involved and high I/O overhead may be incurred. On the other hand, no ranking mechanism is provided to capture the semantic relevancy of the returned result.

Geo-ranking mechanisms were proposed in [8], [2] to assign higher ranking to web pages that are popular among

local users. In these work, a similar strategy with PageRank was proposed to measure the local popularity using back link locations. To further emphasize the local importance, geographic power and spread measurements are defined in different context. Geographic power refers to the popularity of a page in a local area and is measured by the normalized number of desired links to the page. Spread measures how uniform are the distribution of page's back links. The back links of the resources are however not available in most cases. As such, a new geo-ranking mechanism is required to measure the relevance of geo-tags to the area that they locate in.

B. The *mCK* query and relevant query processing

In our earlier work [18], we proposed *mCK* query for finding *m* closest keywords in the spatial database. To answer *mCK* query, a hybrid index based on the R*-tree, the bR*-tree, was proposed. In the bR*-tree, in addition to the the node MBR, each node in the tree is augmented with additional summary information: keyword bitmap and keyword MBR. Keyword bitmap is bitmap structure indicating which keywords are contained in the node and for each keyword, the keyword MBR gives the minimum bounding rectangle that bounds all resources that are associated with the keyword. This information provide a quick summary of the keywords and how they are distributed within the node. Thus, compared to the R*-tree, the bR*-tree has better pruning power due to this additional information being stored.

The challenge of answering *mCK* query is the exponential search space that contains all the combinations of different keywords. Given an initial relatively small distance δ^* , the proposed search process starts from the root node and traverses down the tree in a depth-first manner in order to find a good result as soon as possible. Upon reaching a leaf node, all the objects are exhaustively checked based on the combinations of different keywords.

In addition, an *apriori*-based strategy to search inside one node or across multiple nodes is applied. Two monotonic constraints, namely the distance mutex and keyword mutex, were proposed as the *apriori* properties to be applied in the search process. The two properties have the formal definitions as follow:

Definition 3 (Distance Mutex [18]): A node set \mathcal{N} is distance mutex if there exist two nodes $N, N' \in \mathcal{N}$ such that $dist(N, N') > \delta^*$.

Definition 4 (Keyword Mutex [18]): Given a node set $\mathcal{N} = \{N_1, N_2, \dots, N_n\}$, for any *n* different query keywords $(w_{q_1}, w_{q_2}, \dots, w_{q_n})$ in which w_{q_i} is uniquely contributed by node N_i , there always exist two different keywords w_{q_i} and w_{q_j} such that $dist(w_{q_i}, w_{q_j}) > \delta^*$, then \mathcal{N} is called keyword mutex.

These two properties have been proven to be monotonic and can be used for efficient pruning. Although the experimental results [18] show remarkable scalability in terms of *m* when

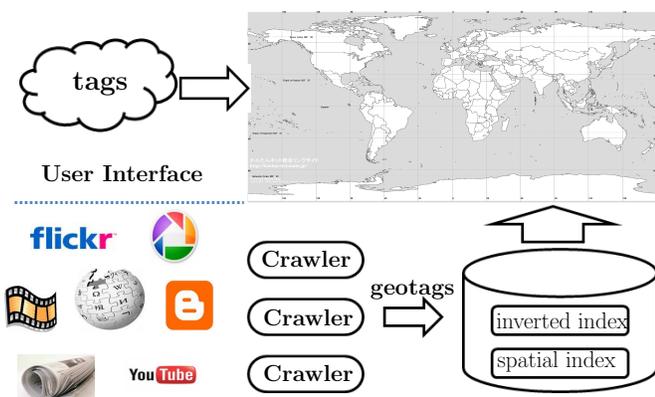


Fig. 2. The framework of location detecting in Web 2.0 applications

answering the mCK query, such an index suffers from high I/O cost if the total number of keywords is large. If there are N keywords, the root node has to maintain N keyword MBRs. The internal nodes become overloaded and occupy a large amount of storage space. Such an index structure prevents the search algorithm from handling spatial database with massive number of tags. In this paper, we propose a new light-weight index based on R^* -tree and inverted index to reduce the I/O cost.

III. DESIGN OF THE SEARCH ENGINE

A. Framework

The emergence of Web 2.0 has resulted in the concept of mashup where data objects from multiple external sources are combined to create a new service. In other words, mashup is a hybrid web application built on top of resources from different channels. In this section, we describe a new type of mapping mashup that integrates data sources from various Web 2.0 applications to support resource locating. We are currently building such a mashup framework within the context of our Marcopolo system [6]. The framework consists of:

- A wide range of data sources from different Web 2.0 applications are combined.
- All the resources are represented in a uniform data model to facilitate the indexing and searching process.
- A simple, map-based user interface is designed to provide users with satisfactory experience in searching and browsing the geographical resources.

As shown in Figure 2, the system is designed to for users to conveniently search and browse the resources. The framework consists of three components: 1) the index engine to crawl, integrate and index source data. 2) query engine to find the location of the resources. 3) friendly interface to improve user experience.

1) *Index Engine*: The index engine aims to support efficient tag matching on top of the combined data resources derived from other applications, such as Wikipedia, Flickr, Picasa Web Albums, and Youtube. In Wikipedia, most of the articles with geographical context have been geo-located. For instance, the page “Forbidden City” is associated with coordinates

$39^{\circ}54'53''N$ and $116^{\circ}23'26''E$. This location information can be parsed as the spatial attribute of the article. In online photo sharing applications like Flickr and Google Web Picasa, APIs have been published to access the public albums, photos, tags as well as their locations. Similarly, when users share their videos in Youtube or other online video sharing websites, they may mark in the map the location where the video was shot. Hence, we are able to retrieve mapped resources of different types to build our underlining database.

After the retrieval step, we need to combine the data sources. We can use the uniform resource mapped resource model to seamlessly integrate the articles, photos and videos. It can provide a transparent access layer and benefit the indexing and searching process. Our index structure is based on R^* -tree and inverted list.

2) *The Query Engine*: Based on our uniform resource model, the query interface allows users to submit query tags to find matching co-located geographical resources. The query input is a collection of the tags which are either associated with the resources or created by users to identify relevant areas in the map. For example, “wedding church New York” is a query to find the churches in New York that can hold a wedding ceremony. Local search engine can benefit greatly from such queries as these can help them to provide better customized service.

Meanwhile, the query engine also supports resource locating when a search boundary constraint is specified. The area can be either added by users manually or set as the current display region in the map. Since the underlying index is actually inverted list, all the elements that do not lie in the query region will be eliminated to ensure all the candidates are within the boundary.

3) *User Interface*: The search engine is designed to be simple and friendly. The whole interface is map-based. Users can freely explore geo-resources in the area they are interested in. The main search entry is only a simple textbox to accept query tags. All the related resources will be displayed directly in the map for further refinement.

Users are also allowed to specify the search area and resource type during the search. They can specify the search constraint by drawing a rectangle in the map and choose the desired resource type they are looking for. The resources that are outside the search boundary or not compatible with the query resource type will be filtered.

B. Light-weighted Index Structure

In Web 2.0 context, users can continuously contribute new resources to the map. On one hand, there will be increasing number of locations. On the other hand, in each location, the number of associated tags will increase as well. Thus, it is essential for the proposed index structure to be scalable enough to handle large number of locations and tags. To achieve this goal, we do not maintain the additional summary information. Moreover, we do not integrate all the locations and tags into one tree structure. Instead, we split them into two components: a spatial index and an inverted index, as shown in Figure 3.

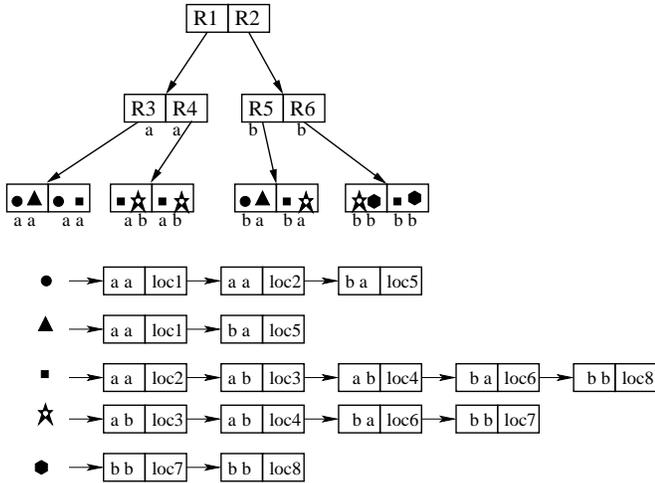


Fig. 3. The index of R*-tree and inverted index

This also ensures that the proposed index could be grafted into existing commercial systems easily.

The R*-tree is used to index all the spatial locations associated with tags. It is constructed in the same manner as described in [4] except that each node is assigned a label indicating the path to the root node. In our example, node R_3 and R_4 are both labelled as “a” because they share the same path to the root node’s first entry. Similarly, R_5 and R_6 are labelled with “b”, which represents the second entry in the root node. Given a node label, we can judge where the node is located in the R*-tree without accessing the tree. Two locations close to each other probably have the same prefix of node label. If they lie in the same internal node, they will be assigned the same label. Thus, the label can be used to approximate the spatial distance between the data points.

An inverted index is built along with the R*-tree. It maintains inverted lists for all the tags in the database. Each element in the list consists of the node label derived from the construction of the R*-tree and the actual location. Note that the list of locations are ordered by the label so that the data points close to each other in geographical space are probably still close in the inverted lists.

Such an index is scalable in terms of both the number of locations and tags. Each time a new location is marked in the map, it is inserted in the labelled R*-tree. The function *ChooseSubtree* in [4] is first invoked to find a leaf node to accommodate the location. If there exist empty entries in the node, the location point is inserted and assigned with the node’s label. The inverted lists of the associated tags can also be updated at a small cost as the elements have been ordered. Otherwise, split occurs in the overflowing node and the changes are propagated upward the tree. Besides the MBR adjustment, we need to update the label of nodes as well as the elements in the inverted lists assigned with the old label. Suppose the propagation stopped at node N labelled $l_1 l_2 \dots l_t$, all the descendent nodes of N will be re-labelled. Meanwhile, the elements in the affected inverted index whose labels start with $l_1 l_2 \dots l_t$ are also updated. Since the lists have

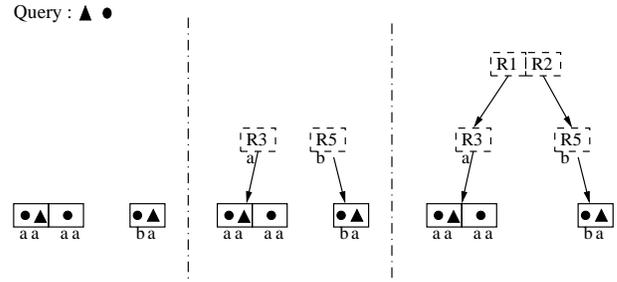


Fig. 4. Bottom-up construction of virtual bR*-tree

been ordered, it is convenient to retrieve the list segment with this prefix. However, if the insertion occurs frequently, ensuring correct label will be computationally expensive. As we do not require accurate labelling in our search algorithm, we can adopt a lazy approach in which we will delay the updates of the labels. The location’s label is buffered before the split operation. The affected part of the R*-tree will be updated in a batch manner using the buffered information to ensure an acceptable cost. The case of insertion of a new tag is much simpler. A new inverted list is created for the new tag and its location and label are inserted into the list.

In contrast to [18], such a light-weight index saves a large amount of I/O cost compared to indexing all the tags and locations into one bR*-tree. Given a set of query tags, only the relevant location lists will be retrieved. The following subsection will introduce how the *mCK* query can be answered on top of the inverted lists without even accessing the spatial index. The index can also be utilized to answer queries in which the user specifies a bounding region. We traverse down the R*-tree as much as possible while ensuring that the node MBR bounds the query region completely. Using the label of such a node, we filter off all data points that are not prefixed with the label. The retrieval cost is small as the labels in the lists have been ordered. Then, a further check is performed to see if the data point is within the query region so as to obtain the correct result.

C. Bottom-Up Search Algorithm

Given m query tags, we retrieve m lists of data points that match the tags from the inverted index. A naive solution to this problem is to exhaustively examine all possible sets of m tuples from different lists. This is prohibitively expensive when the number of objects and/or m is large. To take advantage of the spatial information embedded in the label, we instead propose an elegant solution to construct a virtual bR*-tree using the label and location of the data points.

The virtual bR*-tree is built level by level in bottom-up manner as illustrated in Figure 4. At first, m inverted lists corresponding to the query tags are retrieved and merged into one list ordered by the node label. The cost of this merge sort is linear to the size of lists. We traverse the sorted list and fetch all the data points with the same label. These points are used to construct a new virtual node which has a counterpart in the original R*-tree. Note that the MBR of the virtual node

is much smaller than its counterpart as it is built on the points relevant to the query. For each virtual node, we maintain the additional information: the keyword bitmap and the keyword MBR to summarize the keywords and their distribution inside the virtual node. Compared to the bR*-tree proposed in [18], the node size has been greatly reduced to save the I/O cost and allows the virtual bR*-tree to handle a database with massive number of possible tags. In addition, the a priori-based search strategy can still be applied in the virtual bR*-tree.

Algorithm 1 Bottom-Up Search Strategy

Input: m query tags, inverted index

Output: Distance and location of m closest keywords

1. Retrieve m inverted lists for each query tag
 2. Merge the lists into one list L ordered by the label
 3. Initialize a virtual node cur_node
 4. **while** $L.level < tree.height$ **do**
 5. **for** each element $vnode$ in L **do**
 6. **if** $vnode$ has the same label with its previous element **then**
 7. add $vnode$ into cur_node
 8. **else**
 9. SubsetSearch(cur_node)
 10. add cur_node to List L'
 11. move L' to L
-

The detailed search algorithm is shown in Algorithm 1. Each time a virtual node is constructed, it will be treated as a subtree and the pruning algorithm SubsetSearch proposed in [18] could be applied in this virtual node. The difference is that the search space will exclude the single child node that matches all the query tags. As our search strategy is bottom up, the space within the node must have been explored. Fig 5 shows the order of *NodeSet* candidates checked in the top-down and bottom-up search algorithm respectively. The bottom-up strategy can access the leaf nodes earlier than top-down method and get a smaller δ^* first. In overall summary, the bottom-up search demonstrates better scalability because it only accesses a small virtual bR*-tree and in the meanwhile preserves the effective pruning strategy.

| Order | 1 | 2 | 3 | 4 | 5 | 6 |
|-----------|-------|-------|----------|----------|----------|----------|
| Top-down | R_1 | R_3 | R_2 | R_5 | R_1R_2 | R_3R_5 |
| Bottom-up | R_3 | R_5 | R_1R_2 | R_3R_5 | | |

Fig. 5. Order of *NodeSet* candidates checked

IV. RANKING

The results returned by the mCK search algorithm only considers the spatial closeness while ignoring the geographical relevance. In this section, we propose a new ranking mechanism, namely *geo-tf-idf*, which extends the classic *tf-idf* ranking to be applied in a geographical context.

tf-idf[3], [15], [11] has been widely adopted in search engines to measure the importance of a keyword with respect to a document in a collection or corpus. Intuitively,

$score(k, D)$ will be assigned a higher value if keyword k occurs frequently in document D and infrequently in other documents. Formula 1-5 shows the ranking mechanism with normalization of document length and frequency taken into account:

$$score(Q, D) = \sum_{k \in Q} weight(k, Q) * score(k, D) \quad (1)$$

$$score(k, D) = \frac{tf(k, D)}{dl} * idf \quad (2)$$

$$tf(k, D) = 1 + \ln(1 + \ln(freq(k, D))) \quad (3)$$

$$dl = (1 - s) + s * \frac{dl(D)}{avgdl} \quad (4)$$

$$idf = \ln \frac{N}{df + 1} \quad (5)$$

The term weight of k with respect to Q is usually measured by the raw term frequency in Q . The documents with higher ranking scores will be considered as more relevant to the keywords. Similarly, we can define our score function of a geographical area R with respect to query Q , as shown in Formula 6.

$$score(Q, R) = \sum_{k \in Q} weight(k, Q) * score(k, R) \quad (6)$$

The problem becomes how to measure the importance of a tag k with respect to an area R . Inspired by the intuition behind *tf-idf*, we propose an extended ranking mechanism used in geographical context. A higher score will be assigned to $score(k, R)$ if the tag t and area R satisfy the following properties:

- 1) The tag t appears frequently around the area of R . For example, tourist travelling in Beijing will probably take a visit to Forbidden City. There will be many photos and blogs tagged with “Forbidden City” uploaded in the map. Thus, this tag is closely related to Beijing city.
- 2) The tag t is not frequently mentioned in areas other than R . Although “Forbidden City” may appear in other travel blogs not located in Beijing, such cases are typically rare. Beijing will be assigned with a high score with respect to “Forbidden City”.

In IR systems, the information unit is a document. While in our uniform data model, the concept of document is vague. Each location point is associated with a set of tags. Distinct resources located at the same point can constitute a large virtual document with the tags merged. If there are no resources in the nearby area, we can apply traditional *tf-idf* to measure the weight between keyword k and location p :

$$inw(k, p) = score(k, D) \quad (7)$$

,where D is the merged tag “document” located at point p . However, in real Web 2.0 applications, the resources are contributed and uploaded by users. The resources on the same topic will gather around the actual location. The score value on

a point location p can not completely capture the geographical context. We need to take into account the nearby resources.

In order to measure the effect of tag t in location p to its nearby areas, we build a degradation model as shown in Figure 6(a). The keyword will affect mainly the nearby regions. The regions far away are considered irrelevant to the tag. We may use Gaussian function in Figure 6(b) to describe such degradation. Suppose the d -dimensional space has been normalized into $[0, 1]^d$, the relevance of keyword k located at p with respect to area R can be measured via the following formula:

$$score(k, p, R) = \frac{\int_{q \in R} \{inw(k, p) * f(dist(p, q))\}}{1 + \ln(area(R))} \quad (8)$$

, where f is the degradation function. The intuition behind the definition is that if the area is small and close to the keyword, a higher score will be assigned. If the area is large, the effect of the keyword on the whole region will be scaled down accordingly as well. Thus, R_2 is more relevant to k than R_1 in example of Figure 6(a). If there are multiple occurrences of keyword k in the spatial database SD , the final score will sum up all the effect of k on the area of R :

$$score(k, R) = \sum_{p \in SD} score(k, p, R) \quad (9)$$

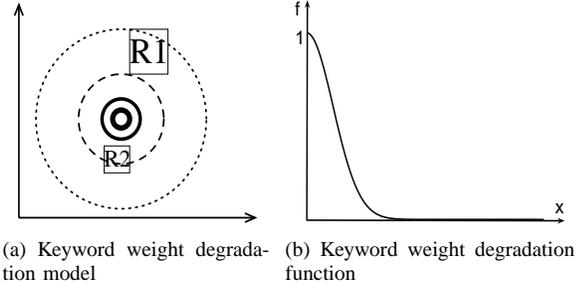


Fig. 6. Degradation of keyword spatial importance

A. Approximate Ranking Mechanism

In real applications, it is expensive to calculate the exact weight of a region with respect to a keyword. To save the computation cost, we propose an approximate scoring mechanism.

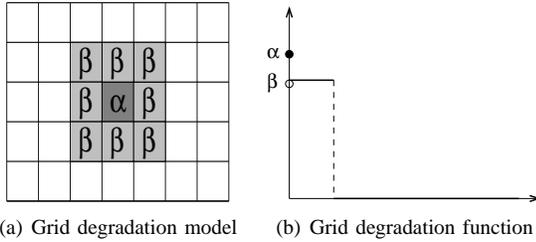


Fig. 7. Degradation of keyword spatial importance

In our approximation model, the space is split into grid cells and region R is approximated by a minimum set of cells that

can bound it. The degradation function no longer decreases continuously to 0. Instead, as shown in Figure 7(b), we use the grid cell as the basic unit. The weight of the grid that holds point p is assigned with constant α and the neighboring grids are assigned with constant β ($1 \geq \alpha > \beta > 0$). The remaining grids are considered not relevant to the keyword. The weight of keyword k with respect the cell C becomes:

$$score(k, C) = \frac{\alpha * \sum_{p \in C} inw(k, p) + \beta * \sum_{C' \in NC} \sum_{p \in C'} inw(k, p)}{3^d} \quad (10)$$

,where NC are the neighboring cells of C . Such a definition will assign higher value to $score(k, C)$ if k frequently appear in C as well as its neighboring cells. Finally, a normalization process similar to idf is proposed to reduce the effect of the general tag that occurs all over the grid:

$$igf(k) = \ln \frac{|G|}{dg(k) + 1} \quad (11)$$

,where $|G|$ is the total number of d -dimensional grid cells and $dg(k)$ is the number of grids containing the keyword. Given all these formulas, the ranking score of k with respect to its location can be approximately defined as :

$$score(k, R) \approx \frac{\sum_{C \in R} score(k, C) * igf(k)}{|C|} \quad (12)$$

,where R is approximated by a set of cells C . Such an approximation takes the nearby documents into account. In our experiments, we will provide further analysis of the ranking mechanism.

V. EXPERIMENT

This section provides an extensive performance study on both synthetic and real data sets in order to evaluate the scalability of our query processing strategy as well as its practical utility.

We incrementally generate large synthetic data sets to simulate those from Web 2.0 applications. Meanwhile, real data sets extracted from online photo sharing applications are used to testify the practical utility of the mCK query in local services and resource locating. All of our experiments are conducted on a server with Quad-Core AMD Opteron(tm) Processor 8356, 128GB memory, running RHEL 4.7AS.

A. Experiments on Synthetic Data Sets

Synthetic data sets are generated to simulate real-life applications in two aspects. First, in successful commercial applications, there are millions of users who created large amounts of resources. Hence, the data size we generated must be large scale. Second, the database expands continuously as new resources are added in by different users. Thus, scalability in terms of the number of locations and their associated tags becomes an essential issue. Our experiments are performed on synthetic data sets with millions of locations and thousands

of tags. All the spatial data points are generated in a d -dimensional space $[0, 1]^d$ in a random manner. For most mapping applications, d is usually set as two. Each data point is randomly assigned with a fix number of tags.

We compare virtual bR*-tree against bR*-tree proposed in our earlier work [18] and MWSJ[13] in answering m CK query. These two algorithms retain the same settings as before. The average response time(ART) is used as our performance metric. In the following experiments, we compare the scalability in terms of m , the number of locations and the associated tags.

1) **Scalability in terms of m :** In this experiment, we randomly generate two data sets with 5,000,000 and 10,000,000 location points respectively. There are in total 5,000 tags in the database and each point is associated with 5 random tags. The number of query tags submitted is varied from 3 to 8.

Figure 8 illustrates the average running time of the three algorithms. The bR*-tree proposed in [18] shows reasonable scalability as m increases. However, since the tree maintains auxiliary information in each node, this lead to extremely high I/O cost. The index occupies more than 4GB disk storage while our inverted index only takes up 527MB for the data set with 5,000,000 points. The performance of MWSJ shows the same pattern as in our previous paper. When m is small, the query can be answered efficiently. When m is increased to larger values, the performance starts to degrade due to the exponential expansion of the search space. Our virtual bR*-tree demonstrates much advantages over the other two algorithms. This is due to the improved index structure as well as the efficient pruning strategy. In the following two experiments, we only compare the performance with MWSJ as the original bR*-tree is not suitable for handling data set with massive number of tags.

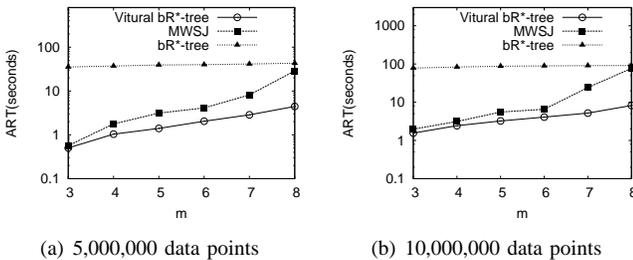


Fig. 8. Scalability in terms of m

2) **Scalability in terms of the number of locations:** In order to simulate the real applications in which new resources are continuously marked in the map, we generate the synthetic data sets with the size increasing steadily from 5,000,000 to 10,000,000 data points. The number of tags associated with each point is fixed as 5. Figure 9 shows the performance trend as the data size increases.

When m is small, both algorithms demonstrate similar growth rate in running time. The spatial index did take effect to suppress the expansion of search space. However, as m increases, the performance of MWSJ becomes sensitive to the growth of data size. The reason is that the search space grows substantially with the data size. A small amount of increase

in data size can lead to remarkable expansion of the search space. In contrast, our virtual bR*-tree is able to scale in a stable manner.

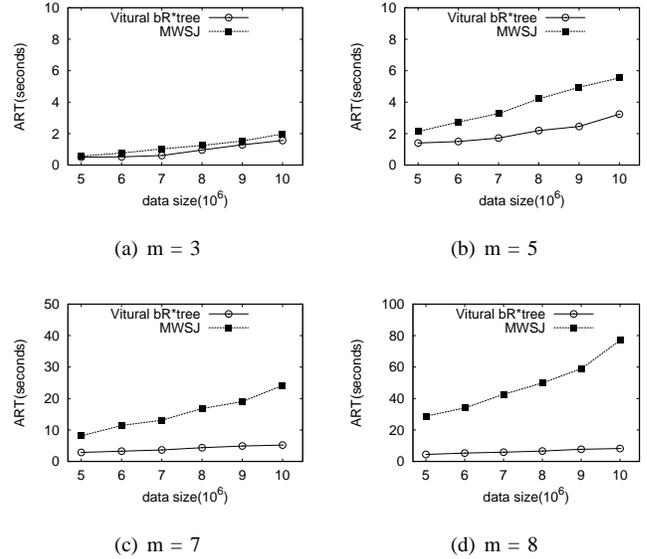


Fig. 9. Scalability in terms of the number of locations

3) **Scalability in terms of the number of tags:** In real applications, new tags can be added to describe a particular resource. In this experiment, synthetic data sets are generated to simulate the growth in the number of tags. Here, we increase the number of tags associated with each location from 1 to 9. The spatial database contains 5,000,000 data points and 5,000 different tags in total.

The two algorithms in Figure 10 present similar growth rate with Figure 9 in term of response time. Their performance are good when m is small. However, MWSJ suffers from serious degradation in handling large number of query tags because it does not inherently support effective summarization of tag locations. Our virtual bR*-tree on the other hand demonstrates good scalability.

Note that the virtual bR*-tree performs slightly better with respect to the growth in the number of tags compared to the growth in number of locations. The reason is that the virtual node in the bR*-tree maintains a bitmap indicating the query tags within. The insertion of a tag into an existing location will only trigger the setting of the bit in the bitmap. However, the insertion of new locations leads to a larger labelled R*-tree. Therefore, more virtual nodes will be created during the search process.

B. Real Experiment

Our real data set is generated via the Flickr service API³ and Picasa Web Albums Data API⁴. We extracted all photos in New York that are tagged and geo-marked. Resources located at the same coordinates are merged into the same tag list. After removing the infrequent tags, the database consists of 74,774

³<http://www.flickr.com/services/api/>

⁴<http://code.google.com/apis/picasaweb/overview.html>

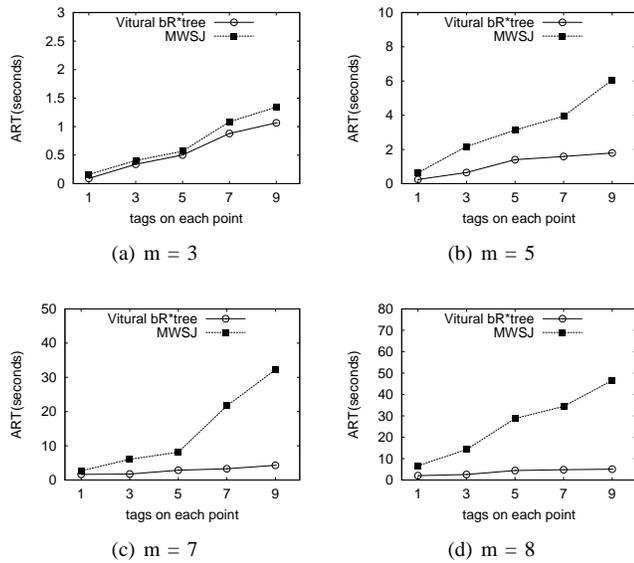


Fig. 10. Scalability in terms of the number of tags

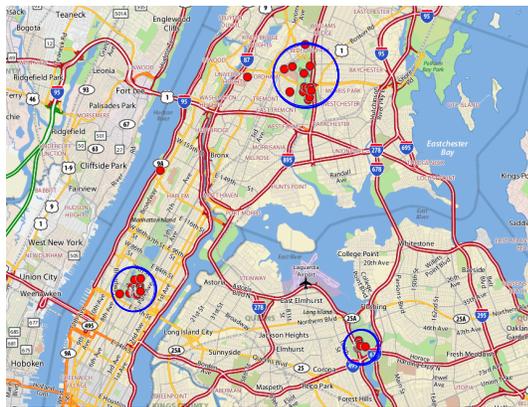


Fig. 11. Distribution of tag “zoo”

location points from Flickr and 6, 729 points from Picasa Web. There are 12, 636 tags in total.

Based on our observation, the geo-tag data set is of acceptable quality. Most of the photos are assigned with relevant tags and are correctly marked in the map. The geo-tags are usually distributed in the form of spatial clusters. These clusters can be utilized to identify the locations of popular resources and events because related tags will emerge around that area. For instance, as shown in Figure 11, the tag “zoo” is mainly distributed in three spatial clusters corresponding to Bronx Zoo, Central Park Zoo and Queens Zoo Wildlife Center respectively. Similarly, in Figure 12 there are a large number of “USOPEN” tags gathering around the Arthur Ashe Stadium where the tennis match is held. This phenomenon provides us with new opportunities to locate resources in more precise geographical scale.

In the following experiments with real data set, we design a set of queries using tags or photos as the input. These queries are mainly for locating local services.

1) *Tag Query*: In this experiment, users propose query tags from the perspective of finding local service, including

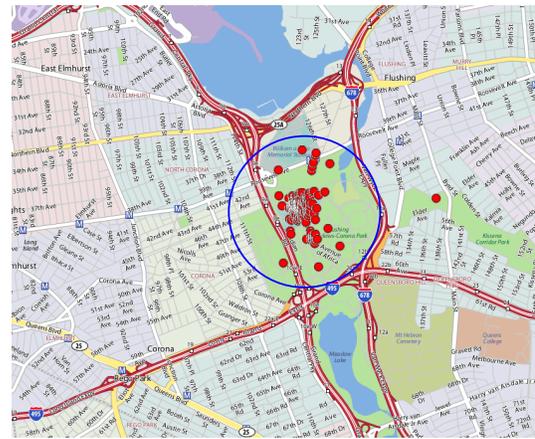


Fig. 12. Distribution of tag “USOPEN”

restaurant, museum, shopping, recreation center, viewing site, local news and so on. As shown in Figure 13, fifteen example queries are listed and the results are compared against Google Maps. These local service queries can be divided into three categories:

| Type | Tag Query | Google | mCK |
|------|---------------------------------------|--------|-----|
| I | <i>liberty statue</i> | ✓ | ✓ |
| | <i>marc jacobs store</i> | ✓ | ✗ |
| II | <i>restaurant seafood sashimi</i> | ✓ | ✗ |
| | <i>bar cocktail jazz player</i> | ✗ | ✓ |
| | <i>museum dinosaur fossil</i> | ✓ | ✓ |
| | <i>tennis court Williams champion</i> | ✗ | ✓ |
| | <i>weapon factory</i> | ✗ | ✗ |
| | <i>river airplane crush</i> | ✓ | ✓ |
| | <i>campus Barack Obama</i> | ✗ | ✓ |
| III | <i>park river fishing</i> | ✗ | ✓ |
| | <i>recreation bowling billiard</i> | ✓ | ✗ |
| | <i>square fountain roller skating</i> | ✗ | ✓ |
| | <i>applestore subway</i> | ✓ | ✓ |
| | <i>supermarket gas station</i> | ✓ | ✗ |
| | <i>hotel church catholic historic</i> | ✗ | ✓ |

Fig. 13. Example tag queries

- The first type of queries is landmark query, such as “Statue of Liberty” and “Marc Jacobs Store” in our examples. Google Maps can answer this kind of queries effectively as enormous number of landmarks have been correctly maintained in the database. Each time a query is submitted, it will first look for gazetteer terms so as to reduce the search space. Our mCK query strategy can correctly give the result for “Statue of Liberty” because this is a famous viewing site in New York City and many photos have been tagged and marked around the statue. However, no result is returned for “Marc Jacobs Store” as no geo-tagged photos about “Marc Jacobs” exist in our data sets.
- The second type of query is seeking for a subject associated with constraint features, persons or events that users are interested in. If the association is common and straightforward, such as a restaurant with seafood and sashimi, a museum with dinosaur fossil and recreation

center with bowling and billiard, Google Maps is a good choice. However, this search engine is not suitable for locating subjects with complicated features like “tennis court where Williams won the champion” and “bar with cocktail and jazz player”. In contrast, our *mCK* query can answer most of the queries as long as the target location is well tagged. For the infrequent tags in our data set, such as “sashimi”, “recreation”, “billiard”, it is difficult for them to occur simultaneously with other query tags in the same location. Thus, no related results are returned. The other problem with *mCK* query is that it can not guarantee that the results returned are related to the same subject. In the query “weapon factory”, Knitting Factory, a music club and concert house, is returned. The reason is that there are no available weapon factories in the database and it happens that there is a poster about weapon on the door of the music club so that the two tags “weapon” and “factory” become connected.

- The third type of query differs in that spatial constraint is embedded. For example, “applestore subway” aims to find the apple retail stores near the subway. Similarly, “supermarket gas station” intends to find a supermarket and gas station close to each other. Google Maps is able to answer these two queries because the result returned happens to contain all the query tags. It can not capture the spatial constraint so as to answer queries like “find a historic catholic church with hotels nearby”. Our *mCK* in essence is proposed to answer this type of query. The quality of the search result relies on the quantity of tags contributed by users.

Figure 14 illustrates some results of example queries returned by *mCK* and Google Maps. We can observe that Google Maps pays more attention to tags with geographical context, such as college and church. The other keywords used as features or spatial constraints are ignored. Thus, its results can not capture the correct query subject or user’s intention. Our *mCK* query takes all the tags into account and find a location matching all of them. Such a query mechanism is able to capture the complete meaning and return satisfactory results.

2) *Ranking Mechanism*: In this part, we provide more in-depth analysis of our ranking strategy. As mentioned, we assign higher value to $score(k, R)$ if keyword k appears frequently around R and infrequently in other locations. Such keywords usually refer to the distinguishing and prominent entities. To achieve this goal, the primary issue is to determine the size of the grid cell. The setting of this parameter is closely related to the specific services being provided. Precise location at the level of a shop or sculpture desires a small cell size. Otherwise, we can allow larger cells to save maintenance cost. Note that hierarchy grid structure can be designed to support locating services at different geographical scales. In our experiment on the New York data set, the grid is split into 500×500 cells.

Fig 15 shows the ranking scores of query tags with respect to the detected location. The bold tags “crash”, “catholic” are

important and prominent subjects or features in that location. Although tags like “river” and “hotel” also appear frequently, they are not distinguishing enough. These tags spread round the city and the *igf*(inverse grid frequency) takes effect to assign lower scores to them. In addition, we can tell from the figure that most feature tags are assigned with moderate scores.

| | | | | |
|----------|---------------------|-------------------------|----------------------------|---------------------|
| t_8 | airplane 141.221 | crash 217.448 | river 118.497 | |
| t_{15} | hotel 114.849 | church 214.979 | catholic 685.173 | historic 45.2946 |

Fig. 15. Ranking score for the tags

The last essential issue about ranking is the weights of the query terms. In default, the query tags are assigned with equal weights. When the results returned are not satisfactory, users are allowed to adjust the weights to highlight the important terms to better identify their intention. For example, given query tags “fountain, square”, a famous square with a fountain may be returned as the square may be frequently tagged, leading to a dominating score. If the user is actually searching an ornate fountain in a square, he can increase the weight of “fountain”. If the fountain is the prominent scene, it is likely to be more frequently tagged than the locating square and the famous fountains will be returned.

3) *Accuracy*: To further test the accuracy of the resource locating, we invite a group of volunteers to generate the local service queries for us and verify whether the returned results are satisfactory or not. Since our data set is still too small to meet with daily needs, we extract frequent tags and ask the volunteers to create the query from the combination of these tags. All the 50 queries are shown in Fig 16.

In this experiment, we first use the Flickr data set and later add in the Picasa Web data set to examine whether the idea of mashup works. The accuracy results are shown in Fig 17. When there is only one data set, *mCK* query only performs slightly better than Google Maps. The reason is that *mCK* seeks for a location matching the query tags but these tags are possible to be associated with different unrelated subjects. If multiple data sources from other applications are combined, it is more likely for the related query tags to appear in the same location and their distance becomes 0. As such, we obtain an improvement in accuracy when the Picasa Web data set is incorporated.

4) *Photo Query*: In this experiment, we discuss how to locate a photo using *mCK* query. Given a photo, the semantic objects as well as their features can be extracted via human intelligence and represented as query tags. These tags are close to each other in the physical space. Google Maps is not suitable to handle such queries because it is unable to capture the spatial constraint. Thus, we try to solve the problem by submitting an *mCK* query using the extracted tags.

As different query tags extracted from the same photo may result in different matching locations, the selection of extracted tags becomes an important issue. Based on our experiments of the query photos in Fig 18, we observe that:

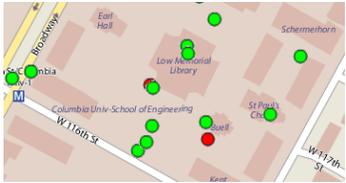
| Query Tags | mCK | Google Maps |
|---|---|---|
| <ul style="list-style-type: none"> ● BarackObama ● campus |  <p style="text-align: center;">Columbia University</p> | <p>Yeshiva University Main Campus</p> <p>Campus Description: 12 acre Yeshiva College campus in uptown Manhattan Stern College campus in midtown Manhattan Sy Syms School of Business at both campuses... a2zcolleges.com</p> |
| <ul style="list-style-type: none"> ● hotel ● church ● catholic ● historic |  <p style="text-align: center;">St Patrick's Cathedral</p> | <p>Old St Patricks Cathedral School</p> <p>Latin Catholic - 2 NYC Cathedrals The people usually known as "Roman Catholics" - the historic Church of the West under the immediate supervision of the Pope ...fordham.edu</p> |

Fig. 14. Example results returned by mCK and Google Maps

| | | | | | |
|---------------------|-----------------------|-----------------|----------------------|----------------------|------------------------|
| sunrise hotel | hospital plaza | dinner movie | island waterfall | weapons factory | girls gold shoes shop |
| wedding church | gold sunset beach | rockband beer | waterfall ferry | weapon museum | 911 monument |
| orange lamp library | historic architecture | historic museum | pizza plaza | sunset lake | japanese restaurants |
| waterfalls hotel | fireworks square | bridge train | river fishing | dogs pet shop | vocation island |
| garden cafe | monument square | movie theatre | kids playground | dinner plaza chicken | ice stadium |
| baseball stadium | sunset skyline | sunset boat sea | tennis usopen | iphone gallery | park band bass |
| beach guitar moon | dance rock band | fish market | fashion jewelry shop | architecture museum | historic movie theater |
| sexy girls | flower exhibition | bronze statue | towers lamp | ice cafe pizza | museum pizza |
| island moon | bowling beer dinner | | | | |

Fig. 16. Local service queries

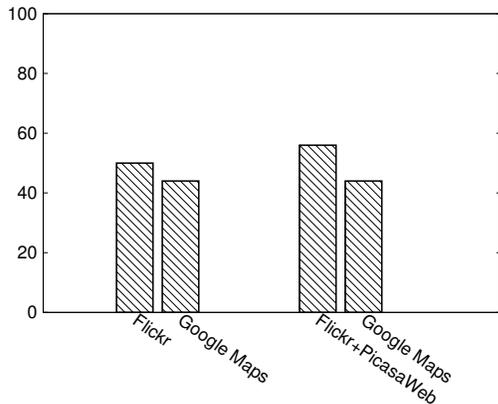


Fig. 17. Accuracy result

tags from the image to eliminate false positive. For instance, in the first photo query, all the four query tags are widely distributed in New York City. Their spatial constraint assists us in detecting the correct location. Missing any of the query tags could lead to a false result.

Other types of resources, such as blogs, news and videos, can also be located in a similar manner. As long as the resource is concerned with a local area, their associated tags are likely to spread around that area. Therefore, mCK is a useful query in the location detection of mapped resources.

VI. CONCLUSIONS

In this paper, we have addressed the new emerging problem of locating mapped resources in Web 2.0. We have proposed to use tags to build a general data and query model to support co-location searches by tag matching. The data resources from different applications can be combined and integrated into the labelled R*-tree and inverted index. Efficient search strategies are developed to effectively answer the tag matching query. We have also proposed a new *geo-tf-idf* ranking mechanism to measure the geographical relevance. Extensive experiments using both synthetic and real life data sets confirm the feasibility and efficiency of our proposed design in the Web 2.0 environment.

- Distinguishing tags are preferred as they can help to reduce the search space. For example, in the second query, "skyscraper" is a frequent tag that appears around the downtown of New York City and leads to many candidates. However, "mast" is a distinguishing keyword as it is found in limited locations. The search space can be further reduced through the spatial constraint that the skyscraper is near the mast.
- The number of candidate locations can be reduced by adding new query tags. When there are no distinguishing features embedded in the photo, users can provide more



Fig. 18. Example photo queries

ACKNOWLEDGMENT

The research and system development reported in this paper was supported by the Singapore National Research Foundation Interactive Digital Media R&D Program, under research Grant NRF2008IDM-IDM004-047.

REFERENCES

- [1] E. Amitay, N. Har'El, R. Sivan, and A. Soffer. Web-a-where: geotagging web content. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 273–280, New York, NY, USA, 2004. ACM.
- [2] S. Asadi, X. Zhou, and G. Yang. Using local popularity of web resources for geo-ranking of search engine results. *World Wide Web*, 12(2):149–170, 2009.
- [3] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.
- [4] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger. The R*-tree: an efficient and robust access method for points and rectangles. *Proc. SIGMOD*, pages 322–331, 1990.

- [5] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The r*-tree: an efficient and robust access method for points and rectangles. *SIGMOD Rec.*, 19(2):322–331, 1990.
- [6] Y. Chen, S. Chen, Y. Gu, M. Hui, F. Li, C. Liu, L. Liu, B. C. Ooi, X. Yang, D. Zhang, and Y. Zhou. Marcopolo: a community system for sharing and integrating travel information on maps. In *EDBT*, pages 1148–1151, 2009.
- [7] G. Cong, C. S. Jensen, and D. Wu. Efficient retrieval of the top-k most relevant spatial web objects. *PVLDB*, 2(1):337–348, 2009.
- [8] J. Ding, L. Gravano, and N. Shivakumar. Computing geographical scopes of web resources. In *VLDB '00: Proceedings of the 26th International Conference on Very Large Data Bases*, pages 545–556, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.
- [9] I. D. Felipe, V. Hristidis, and N. Rische. Keyword search on spatial databases. In *Proc. ICDE International Conference on Data Engineering*, 2008.
- [10] R. Hariharan, B. Hore, C. Li, and S. Mehrotra. Processing spatial-keyword (sk) queries in geographic information retrieval (gir) systems. In *SSDBM*, page 16, 2007.
- [11] F. Liu, C. Yu, W. Meng, and A. Chowdhury. Effective keyword search in relational databases. In *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 563–574, New York, NY, USA, 2006. ACM.
- [12] A. Markowetz, Y.-Y. Chen, T. Suel, X. Long, and B. Seeger. Design and implementation of a geographic search engine. In *WebDB*, pages 19–24, 2005.
- [13] D. Papadias, N. Mamoulis, and Y. Theodoridis. Processing and optimization of multiway spatial joins using R-trees. *Proc. PODS*, pages 44–55, 1999.
- [14] R. S. Purves, P. Clough, C. B. Jones, A. Arampatzis, B. Bucher, D. Finch, G. Fu, H. Joho, A. K. Syed, S. Vaid, and B. Yang. The design and implementation of spirit: a spatially aware search engine for information retrieval on the internet. *Int. J. Geogr. Inf. Sci.*, 21(7):717–745, 2007.
- [15] A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. In *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 21–29, New York, NY, USA, 1996. ACM.
- [16] C. Wang, X. Xie, L. Wang, Y. Lu, and W.-Y. Ma. Detecting geographic locations from web resources. In *GIR '05: Proceedings of the 2005 workshop on Geographic information retrieval*, pages 17–24, New York, NY, USA, 2005. ACM.
- [17] L. Wang, C. Wang, X. Xie, J. Forman, Y. Lu, W.-Y. Ma, and Y. Li. Detecting dominant locations from search queries. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 424–431, New York, NY, USA, 2005. ACM.
- [18] D. Zhang, Y. M. Chee, A. Mondal, A. K. H. Tung, and M. Kitsuregawa. Keyword search in spatial databases: Towards searching by document. In *ICDE*, pages 688–699, 2009.