© 2017 IJEDR | Volume 5, Issue 3 | ISSN: 2321-9939

Improved Framework Of Location Aware Keyword Query Suggestion (LKS) In Geo-Based Social Applications

Klapala Mounika[1], M.Sailaja[2], V. Siva Parvathi[3]
[1]M.Tech (CSE), [2].Assistant Professor, [3] Assistant Professor.
Department of CSE, Prasad V Potluri Siddhartha Institute of Technology, Kanuru, Vijayawada, A.P., India.

***Abstract* — Keyword suggestion in web search helps user to access relevant information without having to known how to precisely express their queries Exiting keyword suggestion techniques do not consider the location of user and the query result the spatial proximity of user to the retrieved result is not taken as a factored in the recommendation. However the relevance's of search result in many application location based services is known to be correlated with proximity to the query issuer. Each query is related to one of topics identified in the conversion fragments preceding the recommendation and is submitted to a search engine over the English we propose in this paper an algorithm foe diverse merging of these lists using a sub modular reward function that reward the topical similar of documents to the conversation words as well as there diversity. We evaluates the proposed method through crowd sourcing the result superiority of the diverse merging technique over Several other which enforce the diversity of topics.**

***Keywords* —Security, Location-Based Social Applications, Location Transformation, Efficiency.**

1. INTRODUCTION

Keyword suggestion (also known as query suggestion) has become one of the most fundamental features of commercial web search engines. The major issue in today's web search is queries entered by users are very short and ambiguous. Users try multiple queries and still don't get the required results because keyword suggested by search engine not very much good descriptor of what actually user needs. Many search engines have made query suggestions techniques for overcoming these problems. Spatial keyword search is very important and will precisely provide keyword suggestions. Location-aware Keyword query Suggestion framework is useful. In this proposed method is based on a query clustering process in which groups of semantically similar queries are identified.

The location aware keyword(LKS) query suggestion method provide the suggested queries retrieve documents which is related to user information and located near to users location. In this paper we will study all the techniques which are used for keyword suggestions to help user to retrieve the correct information.

Before explaining the elements of the framework, we briefly describe the various applications that can be occupied in the framework. The class of applications that take as input spatial keyword queries, processes them and output sources of information that are relevant to the queries are the geographic information retrieval applications [2]. We are able to locate the places having minimum inter-object distance and maximum ratings. Users requesting for the location of the nearest business or service, such as an ATM, restaurant or a retail store can be retrieved efficiently. The user can use the system to reach any address by specifying it in the query.

2. BACKGROUND WORK

Suggestion of keyword is one of the vital characteristic for various search engines. In which, the user submits the keyword to the search engine which then suggests number of keyword query in order to refine the search. But, here the user may not be satisfied with the outcomes, therefore effective methods of keyword suggestion can be used which are based on information obtained from the query logs [10], [11], [12]. New keywords can be recommended on the basis of relation to the main keyword query. The methods which are existing do not provide location - based query suggestion, in which is the keyword suggested, along with the requirement of the user also considers the location of the user and then provides the asked document. The requirement is due to the increased need of spatial keyword search. In 2011, Google processed average of 4.7 billion per day, having huge share of spatial web objects that is the point of interest having web for locations along with their text description or geo -document, the documents related to the geometric locations. Related work on the location – based query suggestion is, the Location – aware Keyword Query suggestion (LKS) framework. In LKS framework, it retrieves keyword that is related to the provided needs of the user information along with retrieving the document that is near to the user location. And hence, it differs from other keyword – aware recommendation methods [9]. The retrieval of geographic information has a way of representing thematic and geographic knowledge within queries. Thematic information is managed by information retrieval method such as keyword matching and geographic information is given by names of places. The place names are then converted to be stored in the spatial database as geometric coordinates. Then the likeliness among the queries is calculated by Euclidean distance or the areas which have overlapping geographic impressions [13].

A direction aware spatial keyword search technique, first considers a collection of Points of Interest (POIs) where every POI is linked with spatial information and textual representation. When a direction aware spatial keyword query is given with a location, direction and a set of keywords, the direction aware search locates the k nearest neighbors of the query which are in the direction of the search and accommodates all the keywords in the input [14]. The existing geo-textual objects help the users receive up-to-date objects whose locations have a spatial overlap with the region specified by the user and the texts consist of the keywords specified by the user.

In systems like these, both the keyword specified by user and the spatial region in a query perform as filters. But such a system has a few problems: 1.The user may receive very few matching geo-textual objects or may receive a huge number of matching objects, which depends on the query region or keywords specified. 2. Specifying the size of spatial region and the query keywords when they are used as filters. To overcome these problems, the ranking-ordering of the geo-textual objects is done which returns only the top-ranked objects[4]. Spatial keyword query is a query that specify both location and keyword set. In spatial keyword query, query keywords are ranked according to their distance from a specified location[2]. For solving a spatial keyword queries an algorithm used spatial keyword search and information retrieval (IR), for that purpose information retrieval (IR2 -Tree), which structure is based on R-Tree is used. Storing spatial and textual information IR2 -Tree proposed as an efficient indexing structure .IR2 -Tree is nothing but combination of signature file and R-tree ,each node of IR2 -Tree contain spatial and keyword information[5].

3. ALGORITHMS

In this section, we introduce a baseline algorithm (BA) for location-aware suggestions (Section 3.1). Then, we propose our efficient partition-based algorithm (Section 3.2).

Baseline Algorithm (BA)

We extend the popular Bookmark-Coloring Algorithm(BCA) [28] to compute the top-m suggestions as a baseline algorithm (BA). BCA models RWR as a bookmark coloring process. Starting with one unit of active ink injected into
node kq, BA processes the nodes in the graph in descending order of their active ink. Different from typical personalized PageRank problems [27], [30] where the graph is homogeneous, our KD-graph Gq has two types of nodes: keyword query nodes and document nodes. As opposed to BCA, BA
only ranks keyword query nodes; a keyword query node retains _ portion of its active ink and distributes 1□_ portion to its neighbor nodes based on its outgoing adjusted edge weights, while a document node distributes all its active ink to its neighbor nodes. In our implementation, the weight of each edge e is adjusted based on _q online, at the time when the source node of e is distributing ink. This means that the edge weight adjustment is done during BA (i.e., Gq needs not be computed and materialized before the algorithm starts).
Moreover, a node may be processed several times; thus, the adjusted weights of its outgoing edges are cached after the node is first processed, for later usage. A node can distribute ink when its active ink exceeds a threshold _. Algorithm BA terminates when either (i) the ink retained at the top mth
keyword query node is more than the ink retained at
the top-(m + 1)th keyword query node plus the sum of the active ink of all nodes [30] or (ii) the active ink of each node is less than _ (typically, _ = 10□5).

Algorithm 1 is a pseudo code of BA. Priority queue Q maintains the nodes to be processed in descending order of their active ink (line 1). Q initially contains one entry, i.e., the user-supplied keywords kq with active ink 1 (line
2). Priority queue C, initially empty, stores the candidate suggestions in descending order of their retained ink (line 1).The sum of the active ink of all nodes AINK is set to 1 (line 3). Termination conditions (i) and (ii) are checked at lines 4 and 8, respectively. The processing of a keyword query node.

Algorithm 1. Baseline Algorithm (BA)

```
   Input: $G(D, K, E), q = (k_q, \lambda_q), m, \epsilon$
   Output: $C$
1  PriorityQueue $Q \leftarrow \emptyset, C \leftarrow \emptyset$;
2  Add $k_q$ to $Q$ with $k_q.aink \leftarrow 1$;
3  $AINK \leftarrow 1$;
4  while $Q \neq \emptyset$ and $Q.top.aink \geq \epsilon$ do
5     Deheap the first entry $top$ from $Q$;
6     $tm$ = the top-$m$ entry from $C$;
7     $tm'$ = the top-$(m+1)$ entry from $C$;
8     if $tm.rink > tm'.rink + AINK$ then
9        break
10    $distratio = 1$;
11    if $top$ is a keyword query node then
12       $distratio = 1 - \alpha$;
13       $top.rink \leftarrow top.rink + top.aink \times \alpha$;
14       $AINK \leftarrow AINK - top.aink \times \alpha$;
15       if there exist a copy $t$ of $top$ in $C$ then
16          Remove $t$ from $C$;
17          $top.rink \leftarrow top.rink + t.rink$;
18       Add $top$ to $C$;
19    for each node $v$ connected to $top$ in $G$ do
20       $v.aink \leftarrow top.aink \times distratio \times \tilde{w}(top, v)$;
21       if there exists a copy $v'$ of $v$ in $Q$ then
22          Remove $v'$ from $Q$; $v.aink \leftarrow v.aink + v'.aink$;
23       Add $v$ to $Q$;
24 return the top-$m$ entries (excluding $k_q$) in $C$;
```

involves retaining _ portion of its active ink (line 13) and distributing 1 □ _ portion to each of its neighbor document nodes based on the adjusted edge weights (lines 19–23).The total active ink AINK is modified accordingly (line 14).As soon as a keyword query node has some retained ink, it enters C. The processing of a document node involves distributing all its active ink to neighbor keyword query nodes according to the adjusted edge weights (lines 19–23). The algorithm returns the top-m candidate suggestions other than kq in C as the result (line 24).

Partition-based Algorithm (PA)

Algorithm BA can be slow for several reasons. First, at each iteration, only one node is processed; thus, the active ink drops slowly and the termination conditions are met after too many iterations. Second, given the large number of iterations, the overhead of maintaining queue Q is significant. Finally, the nodes distribute their active ink to all their neighbors, even if some of them only receive a small amount of ink. We note that existing pre-processing techniques that can accelerate RWR search and BCA (e.g., the pre-selection of hub nodes [28]) require complete knowledge
of the graph before the algorithm starts. Therefore, they are not applicable to our problem, because the edge weights in graph Gq depend on the query location, which is unknown in advance. Applying a pre-computation technique for all possible query locations (i.e., all possible Gq) has extreme
computational and storage requirements.
To improve the performance of BA, in this section, we propose a partition-based algorithm (PA) that divides the keyword queries and the documents in the KD-graph G into groups. Let PK = fPK
i g be the partitions of the keyword queries and PD = fPD i g be the document partitions. Algorithm PA follows the basic routine of algorithm BA, but with the following differences:

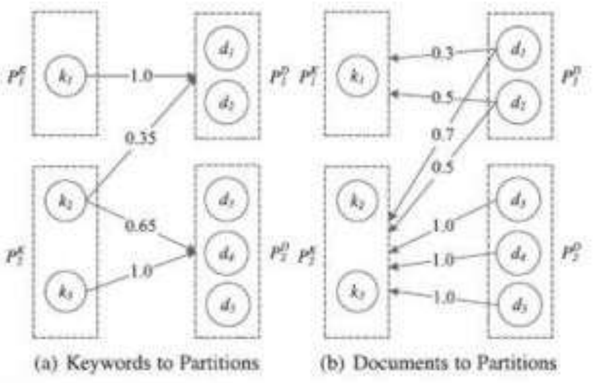


Fig. 4. Node-partition graphs.

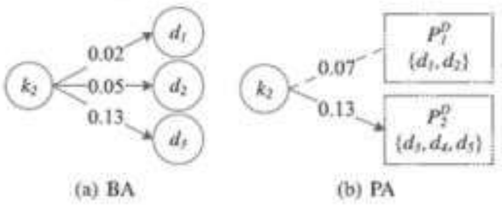


Fig. 5. Ink distribution methods.

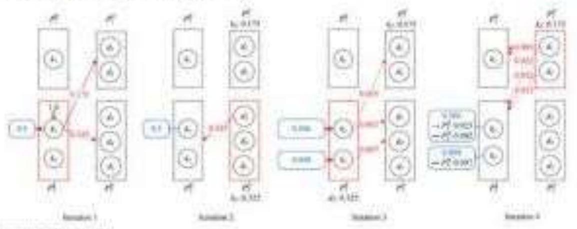


Fig. 6. Illustration of Algorithm PA.

Algorithm 2. PA

```
   Input: $G(D, K, E), G^{KP}, G^{DP}, q = (k_q, \lambda_q), m, \epsilon$
   Output: $C$
1  PriorityQueue $Q \leftarrow \emptyset, C \leftarrow \emptyset$;
2  Add partition $P \ni k_q$ to $Q$ with $P.aink \leftarrow 1$;
3  $AINK \leftarrow 1$;
4  while $Q \neq \emptyset$ and $Q.top.aink_{a_i} \geq \epsilon$ do
5    Deheap the top entry $P_i$ from $Q$;
6    $tm$ = the top-$m$ entry from $C$;
7    $tm'$ = the top-$(m+1)$ entry from $C$;
8    if $tm.rink > tm'.rink + AINK$ then
9      break;
10   Spread the active ink to nodes in $P_i$;
11   for each node $v$ in partition $P_i$ do
12     $distratio = 1$;
13     if $v$ is a keyword query node then
14       $distratio = 1 - \alpha$;
15       $v.rink \leftarrow v.rink + v.aink \times \alpha$;
16       $AINK \leftarrow AINK - v.aink \times \alpha$;
17       if there exist a copy $t$ of $v$ in $C$ then
18         Remove $t$ from $C$;
19         $v.rink \leftarrow v.rink + t.rink$;
20       Add $v$ to $C$;
21       Get partition set $\mathcal{P}$ connected from $v$ in $G^{KP}$;
22     else
23       Get partition set $\mathcal{P}$ connected from $v$ in $G^{DP}$;
24     for each partition $P_i$ in $\mathcal{P}$ do
25       $ink \leftarrow v.aink \times distratio \times \tilde{w}(v, P_i)$;
26       if $ink + v.acc.P_i \geq \epsilon$ then
27         $P_i.aink \leftarrow ink + v.acc.P_i$;
28         if there exist a copy $P_i'$ of $P_i$ in $Q$ then
29           Remove $P_i'$ from $Q$;
             $P_i.aink \leftarrow P_i.aink + P_i'.aink$;
30         Add $P_i$ to $Q$;
31       else
32         Accumulate $ink$ at node $v$ for $P_i$ ($v.acc.P_i$);
33 return the top-$m$ entries (excluding $k_q$) in $C$;
```

TABLE 1
Parameters

Description	Parameter	Values	Default Value
Number of partitions	N	1,4,16,64,256	16
RWR probability	α	0.2,0.35,0.5, 0.65,0.8	0.5
Edge weight adjustment param.	β	0,0.25,0.5, 0.75,1	0.5
Threshold ($\times 10^{-5}$)	ϵ	0.1,0.5,1, 5,10	1
Number of documents (M)	$\|D\|$	0.5, 1, 1.5, 2, 2.5	1.5

Partitioning Methods. We consider four partitioning methods for keyword query/document nodes, to be empirically evaluated in Section 4.

Random Partitioning. Keyword queries are evenly and randomly partitioned into a predetermined number of partitions. The same is done for the documents. Spatial Partitioning. First, a regular grid is used to partition the Euclidean space of document locations. The documents whose locations lie in the same grid cell form a partition. The keyword queries are partitioned according to the document partitions. The idea is inspired by the duality of word and document clustering [31]. Specifically,let PD = fPD 1 ; PD 2 ; _ _ _ ; PD n be the document partitions.We initialize N empty keyword query partitions PK = fPK 1 ; PK 2 ; _ _ _ ; PKN g. According to graph GKP,which connects keywords to document partitions, each keyword query node ki is connected to a set of documents Iteration 1 Iteration 2 Iteration 3 Iteration 4 ment partitions P(ki) = fPDi g. Let PDj be the documentpartition connected from ki with the highest edge weight,i.e., argmaxPDj 2P(ki) w(ki; PDj). Keyword query node ki is added to partition PK j that has the same subscript j as partition PD j . In the end, some of the keyword query partitions in PK might be empty, and thus are removed. Textual Partitioning. Each document d is associated to a vector d: where each dimension refers to a keyword queryk connected to d in the KD-graph G and the dimension is the edge weight w(d; k). Let cos(di: ; dj:) be the cosine similarity between the vectors of documents di and dj . A clustering algorithm, e.g., k-means, is applied on the document vectors using the cosine similarity, so that the document partitions are obtained. The keyword queries are partitioned based on the document

partitions in the same way as in the spatial partitioning method. Hybrid Partitioning. Let fH(di; dj) = _cos(di: ; dj:)+(1□)_dist(di:_; dj:_) be a hybrid distance between documents di and dj , considering both their Euclidean distance and cosine similarity. A clustering algorithm, e.g., k-means,
is applied on the documents using the hybrid distance, so
that the document partitions are obtained. The keyword
queries are partitioned as in the previous methods.

4. IMPLEMENTATION

A. User aware module: This is the module the user can be authenticated whether the user is valid user or not before that the user wants to register first. In registration the user have to give username, password, mail id. If the user is valid the user enters into the application.
B. Domain Specific Search: This module is the simple search engine , when user given a keyword ,the system find out the related documents corresponding to the keyword
C. Location aware keyword query suggestion: In this module the suggestion of a searching query will be display depending upon the latitude and longitude of the user . The location of the particular place will also display in a Google map.
D. Keyword routing Term mapping: In our project maintain a stop word file which contain around 200 commonly seen stop words. The token obtained from the previous step is compared with stop word file. When a match occur corresponding token is removed otherwise it is given as the input to the next stage. Stop word list includes commonly used adjectives, connectives, verbs and certain other words. All the stop-words are removed from the Meta data. Split keyword set: removing all the stop words from the meta data, then the meta data contain only the basic keyword set . these keywords are combine with each other and then retrieve the relevant document. Candidate query graph: this graph is used to find out the minimum distance between each keyword query in the document, minimum distance document is selected and passed to the user. This document must satisfy both the condition, it is more nearer to the user location also semantically relevant to the initial keyword query.

5. RELATED WORK

The related queries are based in previously issued queries, and can be issued by the user to the search engine to tune or redirect the search process. The method proposed is based on a query clustering process in which groups of semantically similar queries are identified. The clustering process uses the content of historical preferences of users registered in the query log of the search engine. The method not only discovers the related queries, but also ranks them according to a relevance criterion. Finally, we show with experiments over the query log of a search engine the effectiveness of the method.

A key factor for the popularity of today’s Web search engines is the friendly user interfaces they provide. Indeed, search engines allow users to specify queries simply as lists of keywords, following the approach of traditional information retrieval systems. Keywords may refer to broad topics, to technical terminology, or even to proper nouns that can be used to guide the search process to the relevant collection of documents. Despite that this simple interaction mechanism has proved to be successful for searching the Web, a list of keywords is not always a good descriptor of the information needs of users. It is not always easy for users to formulate effective queries to search engines. One reason for this is the ambiguity that arise in many terms of a language. Queries having ambiguous terms may retrieve documents which are not what users are searching for. On the other hand, users typically submit very short queries to the search engine, and short queries are more likely to be ambiguous. From a study of the log of a popular search engine, Jansen et al, conclude that most queries are short (around 2 terms per query) and imprecise. Users searching for the same information may phrase their queries differently. Often, users try different queries until they are satisfied with the results. In order to formulate effective queries, users may need to be familiar with specific terminology in a knowledge domain. This is not always the case: users may have little knowledge about the information they are searching, and worst, they could not even be certain about what to search for. As an example, a tourist seeking for summer rentals ads in Chile may not know that the vast majority of such ads in the Web are for apartments in Vina del Mar , a popular beach in the central part of Chile. In contrast, local users may have the expertise to submit queries with the term Vina del Mar , when they are looking for a location to spend their vacations. The idea is to use these expert queries to help non-expert users. In order to overcome these problems, some search engines have implemented methods to suggest alternative queries to users3 . Their aim is to help the users to specify alternative related queries in their search process. Typically, the list of suggested queries is computed by processing the query log of the search engine, which stores the history of previously submitted queries and the URL’s selected in their answers. A central problem that arises in this context is how to model the information needs associated to a query. Some models proposed in previous work represent a query as the set of URL’s clicked by users for the query. This approach have limitations when it comes to identifying similar queries, because two related queries may output different URL’s in the first places of their answers, thus inducing clicks in different URL’s. In addition, as an empirical study shows , the average number of pages clicked per answer is very low (around 2 clicks per query). Our data shows the same. As in traditional document retrieval, in query recommendation one may expect that the ordering in which the queries are returned to the user plays a central role in the quality of the service, even more important than the set of recommendations itself. As far as we know, a problem not yet addressed is the definition of a notion of interest for the suggested queries.

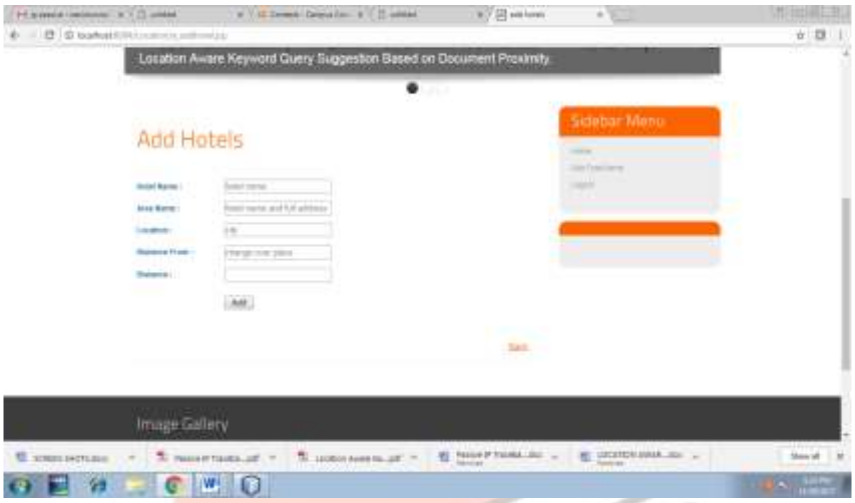
Context-Aware Query Suggestion by Mining Click-Through and Session Data

Query suggestion plays an important role in improving the usability of search engines. Although some recently proposed methods can make meaningful query suggestions by mining query patterns from search logs, none of them are context-aware – they do not take into account the immediately preceding queries as context in query suggestion. In this paper, we propose a novel context-aware query suggestion approach which is in two steps. In the offline modellearning step, to address data sparseness, queries are summarized into concepts by clustering a click-through bipartite. Then, from session data a concept sequence suffix tree is constructed as the query suggestion model. In the online query suggestion step, a user's search context is captured by mapping the query sequence submitted by the user to a sequence of concepts. By looking up the context in the concept sequence suffix tree, our approach suggests queries to the user in a context-aware manner. We test our approach on a large-scale search log of a commercial search engine containing 1.8 billion search queries, 2.6 billion clicks, and 840 million query sessions. The experimental results clearly show that our approach outperforms two baseline methods in both coverage and quality of suggestions.

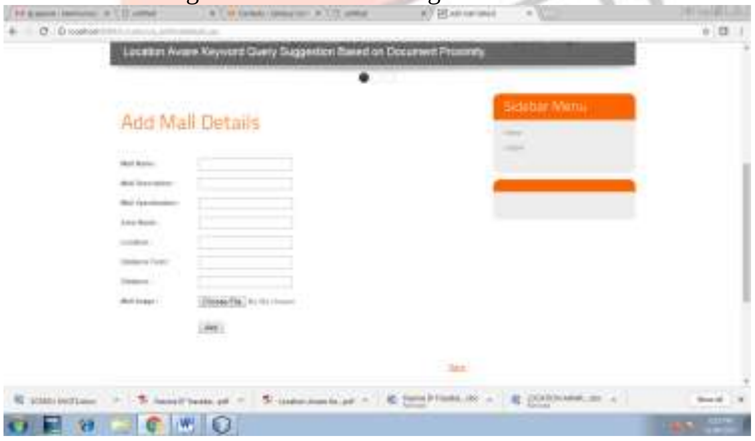
Random Walks on the Click Graph

Search engines can record which documents were clicked for which query, and use these query-document pairs as 'soft' relevance judgments. However, compared to the true judgments, click logs give noisy and sparse relevance information. We apply a Markov random walk model to a large click log, producing a probabilistic ranking of documents for a given query. A key advantage of the model is its ability to retrieve relevant documents that have not yet been clicked for that query and rank those effectively. We conduct experiments on click logs from image search, comparing our ('backward') random walk model to a different ('forward') random walk, varying parameters such as walk length and self-transition probability. The most effective combination is a long backward walk with high self-transition probability.

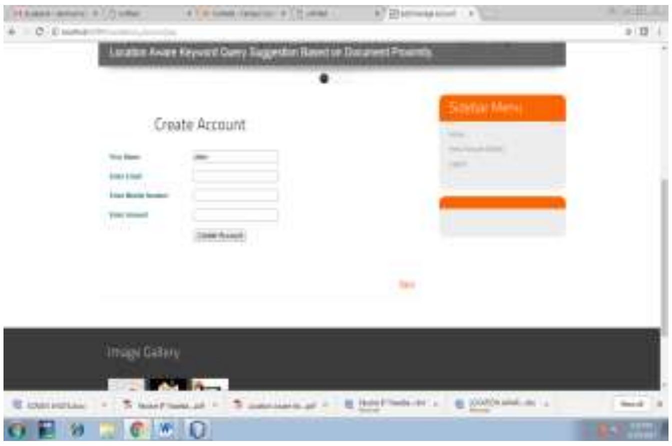
6. RESULTS & PERFORMANCE EVALUTION


Location Aware Keyword Query Suggestion Based on Document Proximity
Add Hotels
Sidebar Menu
Add
Image Gallery

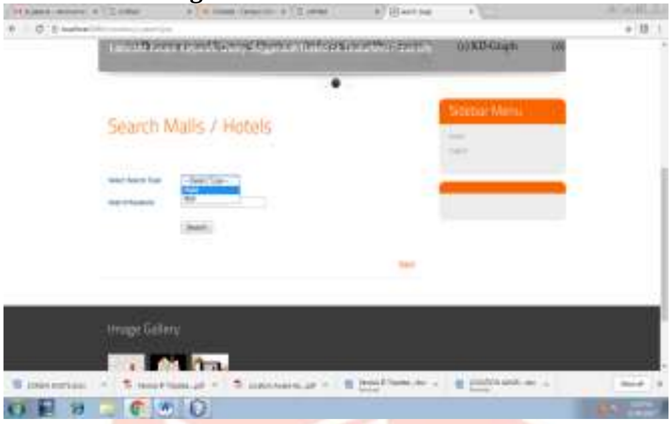
Scr1: admin adding the hotel details along with its room and food details


Location Aware Keyword Query Suggestion Based on Document Proximity
Add Mall Details
Sidebar Menu
Choose File

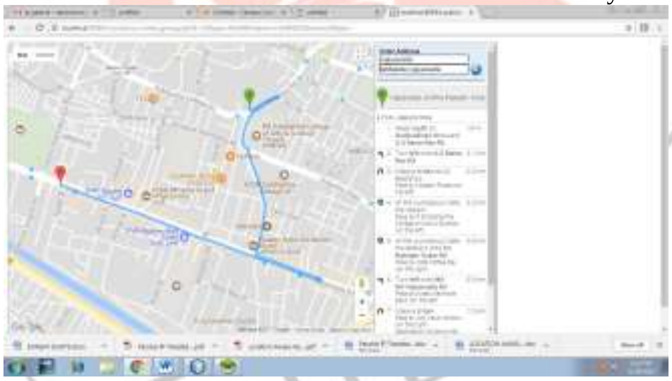
Scr2: admin adding the mall details.


Location Aware Keyword Query Suggestion Based on Document Proximity
Create Account
Sidebar Menu
Create Account
Image Gallery

scr3 : **user creating his account with some amount of balance.**



Scr4: user can search the malls or hotels with a desired keyword.



scr5: user searched hotel or mall location in the gmap.

Performance:
In this section, we evaluate the performance of BA and PA on the two data sets under various parameter settings and report their average runtime (over the workload of 100queries9).**Varying _.** Both BA and PA follow the general idea of BCA [28], and their runtime depends on parameter _. The smaller _ is, the smaller the error incurred by BCA (and BA/PA) on computing the PageRank scores on Gq. We find that when _ _ 10□8, the top query suggestions become stable. Thus, we consider the top suggestions calculated using _ = 10□8 as the base for computing the approximation
error of the top suggestions calculated using _ > 10□8. Following [27], the error is computed as 1 □ AP where AP is the average precision of the retrieved results compared to the true ones. An appropriate _ value should be selected so that (1) the computation can be finished in real time and (2)
the error rate is reasonably low. Figure 11 shows that the response time of BA drops dramatically as _ increases, since a large _ significantly reduces the number of iterations in BA and saves many computations. However, the approximation
error is high for large values of _. On the other hand, PA runs fast even for small values of _, for which the approximation error is low. PA outperforms BA by 1-2 orders of magnitude
when _ is smaller than 10□6. Therefore, under the same time constraint (e.g. < 1s), PA can support smaller _ values, and can thus guarantee a lower error rate. Because of _, BA and PA may produce different suggestions, but the differences are mainly in the low-ranked suggestions. In practice, users only consider the highly ranked suggestions. In our experiments, the top-5 suggestions

offered by BA and PA are identical in 98% and 99% of the times on AOL and TWEET, respectively. **Varying _.** Parameter _ in Equation 3 indicates the random walk restart probability that corresponds to the proportion of ink retained on a query node. As shown in Figure 12 the response times of both BA and PA decrease as _ increases.
This is because for larger _, more amount of ink

9. Experiments with larger workloads gave us similar results.

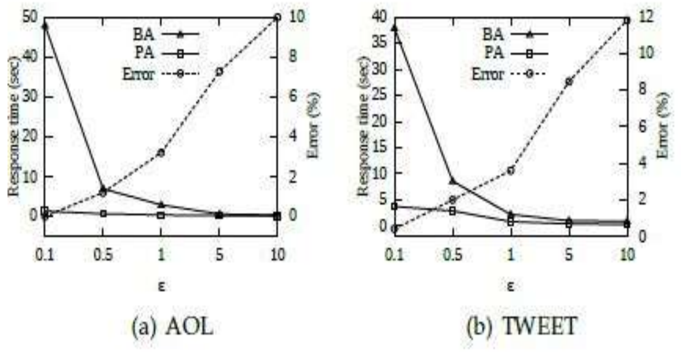


Fig. 11. Response Time when Varying ϵ ($\times 10^{-5}$)

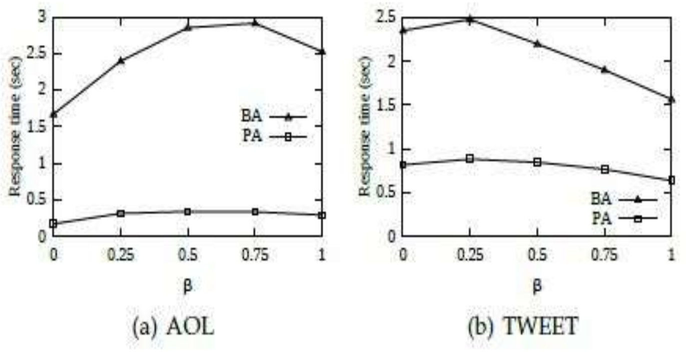


Fig. 13. Response Time when Varying β

is retained at the keyword query nodes; thus, the amount of active ink drops quickly and the termination condition is satisfied early. We observe that PA is more robust to _ and outperforms BA significantly when _ is small. In the special case, when _ = 1, all the ink will be retained on the query node in the beginning, thus both algorithms terminate immediately returning no results. In the other extreme case, when _ = 0, no ink is retained in any node at each step, therefore the ink keeps being redistributed until the random walk process converges to a stable state. In this case, the final scores of the nodes only depend on the structure of the graph, and not on the starting node (query node); no matter what query is given, the suggestions are always the same (i.e., similar to global PageRank scores). Therefore, both extreme cases (_ = 1 or 0) cannot give useful results.

7. CONCLUSION & FUTURE ENHANCEMENT

In this paper, proposed an LKS framework providing keyword suggestions that are relevant to the user information needs and at the same time can retrieve relevant documents near the user location. A baseline algorithm extended from algorithm BCA is introduced to solve the problem. Then, we proposed a partition-based algorithm (PA) which computes the scores of the candidate keyword queries at the partition level and utilizes a lazy mechanism to greatly reduce the computational cost. Empirical studies are conducted to study the effectiveness of our LKS framework and the performance of the proposed algorithms. The result shows that the framework can offer useful suggestions and that PA outperforms the baseline algorithm significantly.
In the future, we plan to further study the effectiveness of the LKS framework by collecting more data and designing a benchmark. In addition, subject to the availability of data, we will adapt and test LKS for the case where the locations of the query issuers are available in the query log. In addition, we believe that PA can also be applied to accelerate RWR on general graphs with dynamic edge weights and we will investigate its general applicability in the future. Moreover, the current version of PA seems to be independent of the partitioning method. It would be interesting to investigate whether alternative partitioning heuristics can further reduce the cost of the algorithm.

8. References

[1] Shuyao Qi, Dingming Wu, and Nikos Mamoulis, "Location Aware Keyword Query Suggestion Based on Document Proximity," Inf. Retr., vol. 16, no. 6, pp. 725–746, 2013.
[2] R. Hariharan, B. Hore, C. Li, and S. Mehrotra, "Processing Spatial-Keyword (SK) queries in geographic information retrieval (GIR) systems," in Proc. 19th Int. Conf. Sci. Statist. Database Manage., 2007, pp. 16–23.
[3] D. Beeferman and A. Berger, "Agglomerative clustering of a search engine query log," in KDD, 2000, pp. 407–416.
[4] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li, "Context-aware query suggestion by mining click-through and session data," in KDD, 2008, pp. 875–883.
[5] N. Craswell and M. Szummer, "Random walks on the click graph," in SIGIR, 2007, pp. 239–246.
[6] Q. Mei, D. Zhou, and K. Church, "Query suggestion using hitting time," in CIKM, 2008, pp. 469–478.
[7] Y. Song and L.-w. He, "Optimal rare query suggestion with implicit user feedback," in WWW, 2010, pp. 901–910.
[8] T. Miyanishi and T. Sakai, "Time-aware structured query suggestion," in SIGIR, 2013, pp. 809–812.
[9] A. Anagnostopoulos, L. Becchetti, C. Castillo, and A. Gionis, "An optimization framework for query recommendation," in WSDM, 2010, pp. 161–170.
[10] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna, "The query-flow graph: Model and applications," in CIKM, 2008, pp. 609–618.
[11] Y. Song, D. Zhou, and L.-w. He, "Query suggestion by constructing term-transition graphs," in WSDM, 2012, pp. 353–362.
[12] D. Wu, M. L. Yiu, and C. S. Jensen, "Moving spatial keyword queries: Formulation, methods, and analysis," ACM Trans. Database Syst., vol. 38, no. 1, pp. 7:1–7:47, 2013.

[13] D. Wu, G. Cong, and C. S. Jensen, “A framework for efficient spatial web object retrieval,” VLDB J., vol. 21, no. 6, pp. 797–822, 2012.
[14] J. Fan, G. Li, L. Zhou, S. Chen, and J. Hu, “SEAL: Spatio-textual similarity search,” Proc. VLDB Endowment, vol. 5, no. 9, pp. 824– 835, 2012.

About authors:

KALAPALA MOUNIKA is presently pursuing M.Tech in Dept of CSE, PVP Siddhartha College of Engineering and Technology, Kanuru, VIJAYAWADA. Her research interests are Data Mining, Networking, Cloud Computing.

Mrs. M. SAILAJA is presently working as a Assistant professor in CSE department, PVP Siddhartha College of Engineering and Technology, Kanuru , Vijayawada.

Mrs. V. SIVA PARVATHI is presently working as a Assistant professor in CSE department, PVP Siddhartha College of Engineering and Technology, Kanuru , Vijayawada.