ISSN:2249-5789
Pippalla Lavanya et al, International Journal of Computer Science & Communication Networks,Vol 7(5),157-161

Efficient Keyword Query Suggestion on Document Proximity Using LKRS

Pippalla Lavanya

M. Tech,
Department of CSE,
Shri Vishnu Engineering College for Women (A),
Vishnupur, Bhimavaram, West Godavari District, Andhra Pradesh.

Ch.Vijaya Krishna

M.Tech, Assistant professor,
Department of CSE
Shri Vishnu Engineering College for Women (A),
Vishnupur, Bhimavaram, West Godavari District, Andhra Pradesh.

Abstract

Keyword and rank suggestion in web search support person to fetch similar spatial location without having to know how to exactly convey their queries. Pervious keyword suggestion mechanism do not taking the locations of the person and the similar query results. The spatial relationship of a person to the similar results is not considered as a factor in the suggestion. However, the similarity of location results in few applications is known to be similarity with their spatial relationship to the query presenter. In my current article, we design a location aware keyword rank query (LKRS) suggestion framework. We suggest a weighted keyword rank document graph, which gets both the similarity between keyword-rank queries and the spatial interval between the resulting documents and the person's location. The graph is browsed in a random-walk-with-restart fashion, to set the keyword-rank queries with the top scores as recommended. To set our current mechanism as scalable, we suggest a partition based approach that outplays the baseline algorithm by up to an offer the magnitude. The appropriateness of the current framework and the achievement of the algorithms are check out real locations data.

Keywords: proximity, LKRS, framework, baseline algorithm, partitioning, graph;

INTRODUCTION

Our current framework, we applied on real world data. Here LTAS (Location aware type a head search) will finds documents nearer to the user location. By using partitioning, baseline, Lq-ranking algorithms support we can fetch good results. Baseline algorithm support we can get the keyword-document relationship results. After that we can take those results, then after we use partitioning algorithm for better result. User will supply query(q), the queries

will form one cluster as q does are return to the user suggestions. For current analysis purpose we use location suggestions based on user logs. User will search particular place and will get the results based on the location suggestion support. Here users pervious search queries that have results close to the current user's related location. Those pervious search results will be stored in log system.

In current article, we are using keyword-query rank suggestion based on user will get location only may not match the current user's search intent with highest rank result. One other hand, our framework aims at suggesting highest rank result that satisfy the person's information needs and have nearby location, based on the authorized person search history.

In current article, our framework support we are applying the rank query suggestion, because they required the relaxed queries to contain the correct results of the current query. By using Random walk computation support we will get similarity location results.

The contributions of this current article are:

We implemented a LKRS framework, which will give suggestions to the similar to the authorized person's information whatever the he needs and can fetch documents close to the authorized query issuer's location with highest rank results. We expand the BCA(Bookmark coloring Algorithm) for Random walk with restart search to evaluate the location-aware recommendation.

IMPLEMENTATION

Authorized person location:-

The authorized person will be authenticated whether the person is valid person or not. Before that the person wants to register initially.

Location aware:-

The search results will be register like hotel name, hotel location, special menu in the hotel and hotel land marks. Authorized person can view the details of the concern hotel or malls information when every the person searches with the help search engine. Here authorized person will find out latitude and longitude of the hotel or mall when ever user give the location of the place.

Authorized User Query:-

In user query module the authorized user will provide a query to find the location in the location finder support. For ex:- the authorized person will provide a name of the hotel or mall name and searching item in location search engine, like hotel or mall name .

Query suggestion:-

The suggestion of the searching query will be exhibits particular place based on the latitude and longitude of the authorized person.

We unitized quick nearest location search to find the nearest place of the authorized person. The location of the particular hotel/mall will also display in Google map.

Current Article Flow

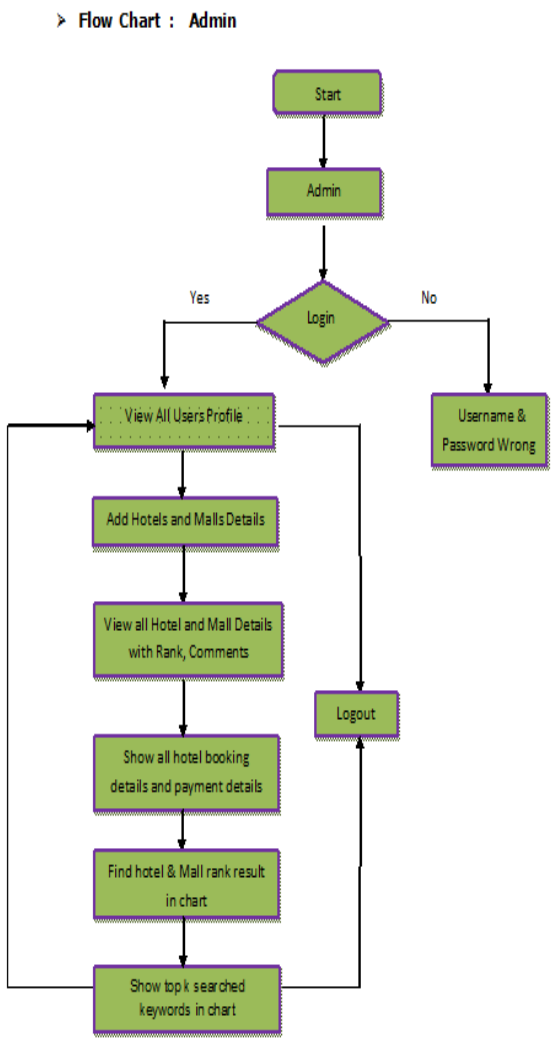

Fig:- Admin flow chart

Authorized person will access his account. Then after admin will view all users details. Admin will add hotels and malls information. He will check all hotel and mall information with rank, comments. Then after he will seen all hotel booking information and payment information. Admin will find hotel and mall rank result in chart format. Then after he will check top k searched keywords in the chart.

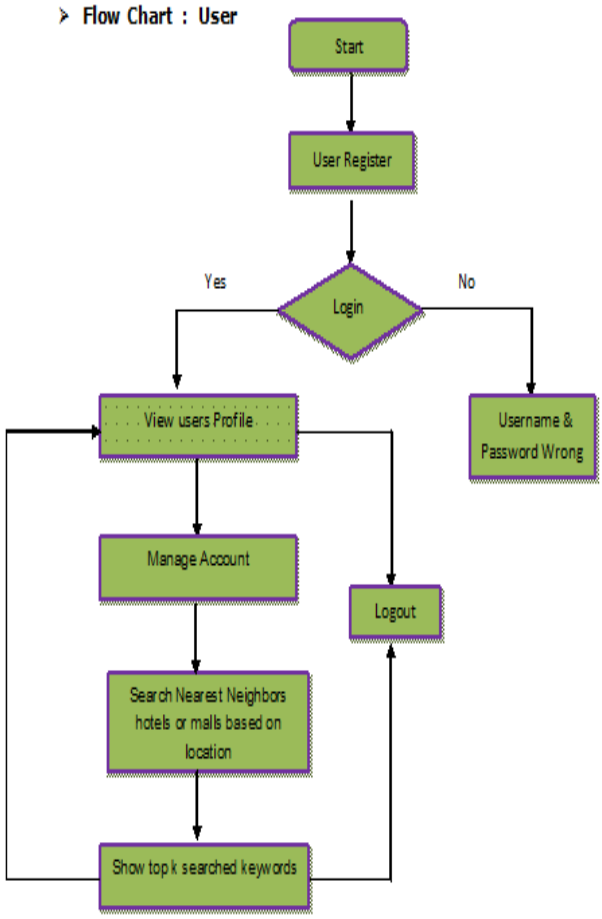

Fig:- user flow chart

First Person needs register to find location. Then next person need to login his own account. Person will view his own account. Person will manage account. He will search nearest neighbors hotels or malls based on location, what every he wants. He will check top k searched keywords.

Algorithm:

ALGORITHM 1: Baseline BA

Input : $G(D, K, E)$, $q = (k_q, \lambda_q)$, m, ϵ
Output: C

```
1  PriorityQueue Q ← ∅, C ← ∅
2  Add k_q to Q with k_q.aink ← 1
3  AINK ← 1
4  while Q ≠ ∅ and Q.top.aink ≥ ε 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 − α
13         top.rink ← top.rink + top.aink × α
14         AINK ← AINK − top.aink × α
15         if there exist a copy t of top in C then
16             Remove t from C
17             top.rink ← top.rink + t.rink
18         Add top to C
19     for each node v connected to top in G do
20         v.aink ← top.aink × distratio × w̃(top, v)
21         if there exists a copy v' of v in Q then
22             Remove v' from Q; v.aink ← v.aink + v'.ain
23         Add v to Q
24 return the top-m entries (excluding k_q) in C
```

Fig:- Baseline(BA)

In our Base line algorithm, we take parameters like Document as 'D', Keyword as 'K' and Edges as 'E'. By using baseline support we can construct graph. Here , we can construct weighted graph with support of baseline algorithm.

ALGORITHM 2: PA

Input : $G(D, K, E)$, G^{KP}, G^{DP}, $q = (k_q, \lambda_q)$, m, ϵ
Output: C

```
1  PriorityQueue Q ← ∅, C ← ∅
2  Add partition P ∋ k_q to Q with P.aink ← 1
3  AINK ← 1
4  while Q ≠ ∅ and Q.top.aink_{v_i} ≥ ε do
5      Deheap the top entry P_t 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_t
11     for each node v in partition P_t do
12         distratio = 1
13         if v is a keyword query node then
14             distratio = 1 − α
15             v.rink ← v.rink + v.aink × α
16             AINK ← AINK − v.aink × α
17             if there exist a copy t of v in C then
18                 Remove t from C
19                 v.rink ← v.rink + t.rink
20             Add v to C
21             Get partition set P connected from v in G^KP
22         else
23             Get partition set P connected from v in G^DP
24         for each partition P_i in P do
25             ink ← v.aink × distratio × w̃(v, P_i)
26             if ink + v.acc.P_i ≥ ε then
27                 P_i.aink ← 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 ← 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
```

Fig:- Partitioning algorithm

Partitioning Algorithm support we can partitioning the keyword graph in to sub graph and document graph in to sub graph. Those sub graph support we can get better results easily.

LQ: RANKING (aLQ, k, q) algorithm

```
Input : aLQ : the aggregate IL-Quadtree, k : number of objects returned, q : the query
Output : R : k objects with highest scores
1 R := Ø; H = Ø;
2 Push root node of the virtual quadtree Q into H ;
3 while H | = Ø do
4 e ? the tuple popped from H ;
5 if e is an object then
6 R := R ? e ;
7 Terminate the Loop if |R| = k ;
8 else
9 if e is not a leaf node then for each child entry e
10 for each child entry edo
11. Compute f( e , q);
12 Push e into H ;
13 else
14 C := Ø ;
15 for each quadtree aLQi where i ? I(q.T ) do
16 e? the black
17 leaf node in aLQi with seq( e ) ? seq(e) ;
C := C? objects in e ;
18 for each object o ? C do
19 Compute f(o, q) and push it to H ;
20 return R
```

Fig:- LQ ranking algorithm

CONCLUSION

We suggest a location based keyword query rank based search that are fetch to the authorized person information need at the same time can fetch semantic similar document nearer to the person location. The similarity search result in few applications like location based services will knows the similarity with spatial relationship to the query provider in Search engine. Authorized person offers complexity in exhibit their web search. Finally we find out the multiple keyword query and location then it will calculate the distance based on the query and location using the quickest search and suggest the results based on authorized person query and nearest to the location with highest rank results suggest.

REFERENCES

[1] M. P. Kato, T. Sakai, and K. Tanaka, "When do individuals utilize question proposal? A question proposal log examination," Inf. Retr., vol. 16, no. 6, pp. 725–746, 2013.

[2] R. Baeza-Yates, C. Hurtado, and M. Mendoza, "Question proposal utilizing question sign in web crawlers," in EDBT, 2004, pp. 588–596.

[3] D. Beeferman and A. Berger, "Agglomerative bunching of a hunt motor question log," in KDD, 2000, pp. 407–416.