eSPAK: Top-k Spatial Keyword Query Processing in Directed Road Networks

Muhammad Attique
Computer Engineering,
Ajou University,
Suwon, South Korea
attique@ajou.ac.kr

Awais Khan
Computer Engineering,
Ajou University,
Suwon, South Korea
awais@ajou.ac.kr

Tae-Sun Chung*
Computer Engineering,
Ajou University,
Suwon, South Korea
tschung@ajou.ac.kr

ABSTRACT

Given a query location and a set of query keywords, a top-k spatial keyword query rank objects based on the distance to the query location and textual relevance to the query keywords. Several solutions have been proposed for top-k spatial keyword queries in Euclidean space. However, few algorithms study top-k keyword queries in undirected road networks where every road segment is undirected. Even worse, insufficient attention has been given to the processing of keyword queries in directed road networks where each road segment has a particular orientation. Therefore, in this paper, we present an algorithm called eSPAK that can efficiently answer the top-k spatial keyword queries in directed road networks. Our experimental results demonstrate that eSPAK significantly outperforms conventional solution in terms of query processing cost.

CCS Concepts

•Information systems → Data management systems; Query processing;

Keywords

spatial keyword queries, directed road networks, location-based services

1. INTRODUCTION

With the popularization of geo-tagged data (e.g., geo-tagged photos, videos, check-ins, and text messages), many online location-based services such as Google Maps, Yahoo Maps, and Bing Maps have started providing useful information via location-based queries [9, 2]. At the same time, a textual description of the point of interests, e.g., hotels, shopping malls and tourist attractions, are easily accessible on the web. These developments call for techniques that efficiently process the top-k spatial keyword queries that return a ranked list of k best facilities based on their proximity to the query location and relevance to the query keywords. Therefore, several algorithms have been proposed for processing top-k spatial keyword queries in Euclidean space [3, 8]. Although few algorithms exist that study the keyword queries in a road network, however, they all focus on the undirected road network. In this paper, for the first time, we are investigating a top-k spatial keyword queries in directed road networks which are more closely related to the real world scenario.

Top-k keyword queries can be used for a wide range of applications in recommendation systems and decision support systems. For example, a tourist may want to retrieve a sorted list of restaurants that serve Italian steak based on shortest distance from her location and textual relevance to the query keywords. Given a set of data objects $D = d_1, d_2, ..., d_{|D|}$, query location and set of keywords, the top-k spatial keyword query returns the best k data objects from D according to their combined textual and spatial relevance to query.

Figure 1 presents an example of a directed road network where rectangles represents the data objects with a textual description, and the triangle represent the query location. The number label on each edge indicates the distance between two adjacent objects e.g., $dist(n_1, d_1) = 1$ and $dist(d_1, n_2) = 2$. Consider a scenario where a tourist is interested in finding an "Italian restaurant". If an undirected road network is considered, the top-1 Italian restaurant is d_6. However, in directed road network the shortest path from q to d_6 is ($q \rightarrow n_3 \rightarrow n_7 \rightarrow d_6$). Therefore, for directed road network, top-1 result is d_3 because it is closer to query location than d_6. Now consider tourist is looking for "Cafe bakery", the data object d_7 may score better than

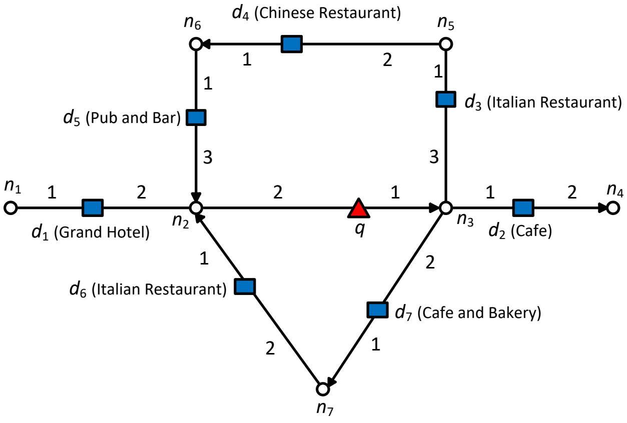

Figure 1: An illustration of directed road network.

[*] Corresponding Author.

the data object d_1 because d_7 ("Cafe and bakery") is more textually relevant to query keywords than d_2 ("Cafe"), and the $dist(q, d_7)$ is just slightly bigger than $dist(q, d_2)$.

Top-k spatial keywords in directed road networks are useful for location-based applications. However, the query processing is costly, because it requires computing several shortest paths while considering the orientation of road segments. To the best of our knowledge, this is the first attempt to study top-k spatial keyword queries in directed road networks. In this paper, we propose a methodology to rank the data objects based on the spatial and textual relevance.

Below, we summarize our contributions:

- We propose an efficient indexing technique that indexes the data objects in inverted files for processing top-k spatial keyword queries in directed road networks.
- We present an algorithm eSPAK that exploits the indexing framework to effectively retrieve top-k results.
- Finally, we conduct extensive experiments on a real road network and demonstrate the superiority of our proposed algorithm over baseline approach.

2. RELATED WORK

Several approaches have been proposed for ranking spatial data objects. Harihran *et al.* [7] proposed an indexing structure KR*-tree by capturing the joint distribution of keywords in space. Ian de Felipe *et al.* [4] proposed a data structure which combines an R-tree with text signatures. Each node of the R-tree exploits a signature to indicate the presence of keywords in the sub-tree of the node. However, both of these approaches only handles boolean keyword queries in Euclidean space.

Top-k spatial keyword queries where data objects are ranked according to their combined textual and spatial relevance to keyword queries was first studied by Cong *et al.* [3] and Li *et al.* [8]. Both studies [8] integrates location indexing and text indexing to generate IR-trees. These studies process top-k spatial keyword queries only in Euclidean space and not suitable for processing top-k spatial preference queries in road networks, where the distance between objects is determined by the shortest path connecting them.

Top-k spatial keyword queries in road networks was introduced by Rocha *et al.* [10]. In particular, they proposed three different indexing techniques (Base Indexing, Enhanced Indexing and Overlay Indexing) for processing spatial keyword queries in road networks. Recently, Guo *et al.* [6] studied continuous top-k spatial keyword queries on road networks. They presented two methods for monitoring moving queries in an incremental manner which reduces the traversing of network edges. Different from [10, 6] in this study we consider top-k spatial keyword queries in directed road networks where each road segment has a particular orientation.

3. PRELIMINARIES

Section 3.1 defines the terms and notations that are used in this paper. Section 3.2 formulates the problem using an example that illustrates the general results of top-k spatial keyword queries.

3.1 Definition of Terms and Notations

Road Network: A road network is represented by a weighted directed graph $G = (N, E, W)$ where N, E and W denote the node set, edge set and edge distance matrix, respectively. Each edge is also assigned an orientation which is either undirected or directed. The undirected edge is represented by $e = \overline{n_s n_e}$ where n_s and n_e are adjacent nodes, whereas directed edge represented as $e = \overrightarrow{n_s n_e}$ or $e = \overleftarrow{n_e n_s}$. Naturally, the arrow above the edge indicates associated direction. We refer n_s as starting node and n_e as ending node of edge. For example in Figure 1, n_6 is starting node of edge $\overrightarrow{n_6 n_2}$, whereas it is ending node for edge $\overleftarrow{n_6 n_5}$. The particular edge where query object is located is called the active edge.

3.2 Problem Formulation

Similar to previous studies [3, 10] we assume each data object $d \in D$ has a point location $d.l$ in road network and a text description $d.t$. Given a query location $q.l$, set of keywords $q.t$ and k number of data objects to return, the top-k spatial keyword query Q_k is defined as $Q_k = (q.l, q.t, k)$, which takes three arguments and returns best k data objects from D according to score that takes into consideration spatial proximity and text relevance. The score $\psi(d)$ of a data object d is defined by the following equation:

$$\psi(d) = \frac{\mu(d.t, q.t)}{1 + \alpha \cdot \lambda(d.l, q.l)} \tag{1}$$

where $\lambda(d.l, q.l)$ is the spatial relevance between $d.l$ and $q.l$, $\mu(d.t, q.t)$ is the textual relevance between $d.t$ and $q.t$ and α is a positive real number that determines the importance of one measure over the other. For example, if only spatial relevance is considered then $\alpha = 0$, if more importance is given to textual relevance then $\alpha > 1$.

Spatial relevance (λ) is defined as the shortest distance between data object d and q: $\lambda(d.l, q.l) = dist(d.l, q.l)$. Thus, $dist(d_i.l, q.l) < dist(d_j.l, q.l)$ indicates that data object d_i is more spatially relevant to q than data object d_j. The textual relevance (μ) can be computed using any popular information retrieval model, such as cosine similarity or language model. In this study, we use the cosine similarity between $d.t$ and $q.t$. The textual relevance is defined as:

$$\mu(d.t, q.t) = \frac{\sum_{t \in q.t} w_t(d.t).w_t(q.t)}{\sqrt{\sum_{t \in p.t}[w_t(d.t)]^2 . \sum_{t \in q.t}[w_t(q.t)]^2}} \tag{2}$$

Here, the weight $w_t(d.t)$ represents the frequency of term t in $d.t$, and the weight $w_t(q.t)$ describes the ratio of total number of data objects in D to the number of data objects that contains t in their description. Higher μ means, the higher textual relevance to query keywords.

4. QUERY PROCESSING SYSTEM

In this section, we present our proposed query processing system that indexes the data objects and prunes the irrelevant edges for efficient query processing. In Section 4.1, we discuss indexing framework and in Section 4.2, we present an efficient keyword query processing algorithm (eSPAK).

4.1 Indexing Framework

We implemented the inverted file for indexing data objects. The inverted file contains vocabulary and inverted lists. The vocabulary keeps general information about each term such as frequency of term which is helpful in computing the textual relevance of the data objects. The inverted list stores the data objects located on the edge $\overrightarrow{n_sn_e}$ that have a term t in their description. An inverted list is identified by a key composed of (e_{id}, t_{id}), e_{id} and t_{id} represents edge *id* and term *id*, respectively. Each inverted file is a set of inverted lists. The separate inverted list is used for each term in the object description. Inverted list stores two attributes for each data object: first, the distance between data object and starting node $dist(n_s, d_i)$; second, the significance factor $\theta(t_i, d_i)$ of the term t_i in the description of the data object. Note that the network distance between two points in directed road networks is not symmetrical (i.e, $dist(n_s, d_i) \neq dist(d_i, n_s)$). Recall that the starting node is chosen according to orientation of edge such that direction of edge is from node towards data object. In Figure 1, n_3 is starting node for d_7. For bi-directional edges any of the adjacent node can act as starting node.

Furthermore, we develop a pruning technique to prune the irrelevant edges. To achieve this, we introduce a highest significance factor (θ_t) of term t in the description of objects lying on the edge. The θ_t on an edge is retrieved by a combination of e_{id} and t_{id}. The θ_t is an upper-bound significance for an object on the edge with t. Naturally, the edges with θ_t smaller than the score of the k-th object found so far are pruned.

The proposed indexing scheme has three main advantages. First, the object search relevant to query keywords is very efficient using the (e_{id}, t_{id}) pair. Second, inverted files also store the network distance between starting node and data object which helps in accessing the data object in the directed road network. Finally, the pruning technique allows faster query processing by exploring fewer edges.

4.2 eSPAK: Query Processing Algorithm

eSPAK traverses the road network incrementally in a similar fashion to Dijkstra's algorithm [5]. Algorithm 1 returns top-k data objects with highest scores according to their joint textual and spatial relevances to the query. The algorithm begins by exploring the active edge where query object q is located and expands the network in an increasing order of distance from q. Each entry in the min-heap takes the form $(p_a, edge)$, where p_a indicates the anchor point in the edge. For an active edge, q becomes the anchor point. Otherwise, for directed edges starting node n_e becomes the anchor point or for bi-directional edges either of the adjacent node, i.e., n_s or n_e becomes the anchor point. Let D_k be the current set of top-k data objects and s_k be the score of *k-th* data object in D_k. $candsearch((e_{id}, t_{id}), s_k)$ function retrieves the candidate data objects D_c located in an edge with better score $\psi(d)$ than s_k. Next, the D_k set is updated with the data objects in D_c and so does s_k. The algorithm continues its expansion and inserts the adjacent edges of the boundary node until the heap is exhausted or the remaining data objects cannot have the better score than s_k.

The $candsearch((e_{id}, t_{id}), s_k)$ procedure finds the candidate data objects in two steps. In first step, the upper-bound score of edges is computed using a significance factor (θ_t) of a term $t \in q.t$ and the shortest distance $sdist(e_i, q.l)$ between edge and the query location. In next step, the inverted lists of term t are fetched, if their upper-bound score is higher than s_k. In inverted lists, the objects whose score $\psi(d)$ is greater than s_k are returned.

Algorithm 1: eSPAK: Query Processing Algorithm.

```
Input: Top-$k$ spatial keyword query $Q_N = (q.l, q.t, k)$
Output: Top-$k$ data objects with highest score
$D_c \leftarrow \emptyset$ /*set of candidate data objects
$max\text{-}heap\ D_k \leftarrow \emptyset$ /*current Top-$k$ set
$s_k \leftarrow 0$ /*k-th score in $D_k$
$min\text{-}heap \leftarrow \emptyset$
$explored \leftarrow \emptyset$
$min\text{-}heap.insert(q.l, edge_{active})$
while $min\text{-}heap \neq \emptyset$ or $(equ)$ do
    $(p_a, edge) \leftarrow min\text{-}heap.pop()$
    if $(p_a, edge) \notin explored$ then
        $explored \leftarrow explored \cup (p_a, edge)$
        $D_c \leftarrow candsearch((e_{id}, t_{id}), s_k)$
        update $D_k$ and $s_k$
    end
    else
        $min\text{-}heap.push(adjacent\ node,\ edge)$
    end
end
return $D_k$
```

In order to give a feel for our proposed algorithm, consider a road network presented in Figure 1. Assume, a query q generated a top-1 keyword query with q.d "Italian restaurant". For the ease of presentation, we assume $\alpha = 1$ and the textual relevance μ is the number of occurrence of query keywords in $d.t$ divided by the number of keywords in the document. For example, the $\psi(d_4) = \frac{\mu(d_4.t, q.t)}{1+\lambda(d_4.l, q.l)} = \frac{0.5}{8} = 0.06$. The algorithm starts the network expansion from active edge $\overrightarrow{n_2n_3}$ where q is the anchor point. Note that the direction of edge $\overrightarrow{n_2n_3}$ is from n_2 to n_3. Therefore, algorithm only explore $\overrightarrow{qn_3}$. There is no data object found in $\overrightarrow{qn_3}$. Then, n_3 becomes anchor point and edges $\overrightarrow{n_3n_4}$, $\overline{n_3n_5}$ and $\overrightarrow{n_3n_7}$ are inserted in min-heap. Next, *candsearch* function retrieves the candidate data objects on edges $\overrightarrow{n_3n_4}$, $\overline{n_2n_3}$ and $\overrightarrow{n_3n_7}$ whose score is better than s_k. On edge $\overline{n_3n_5}$ data object d_3 is retrieved with $\psi(d_3) = 0.2$. The data object d_3 is inserted in D_k set and the value of s_k is set to 0.2. For edge $\overrightarrow{n_3n_4}$ and $\overrightarrow{n_3n_7}$ there is no candidate object found because $d_2.t$ ("Cafe") and $d_7.t$ ("Cafe and Bakery") does not match with $q.t$. The algorithm continues expanding the edges whose upper-bound score is greater than s_k. The edge $\overrightarrow{n_7n_2}$ is explored next, the upper-bound score of $\overrightarrow{n_7n_2}$ is $\frac{1}{7}$ which is less than s_k. Similarly, for edge $\overleftarrow{n_5n_6}$ the upper-bound score is $\frac{0.5}{8} < s_k$. Therefore, algorithm terminates reporting d_3 as top-1 result.

5. PERFORMANCE EVALUATION

In this section, we evaluate the performance of eSPAK through simulation experiments. We describe experiment settings in Section 5.1 and present experimental results in Section 5.2.

5.1 Experimental Settings

All of our experiments are performed using a real road network [1] that comprised the main roads of North America, with 175,812 nodes and 179,178 edges. Both the direction of edges and data points on edges are generated randomly.

Table 1: Experimental Parameter Settings

Parameter	Range
Number of results (k)	5, 10, **15**, 20, 25
Number of data objects |D|	10, 20, **30**, 40, 50 (x1000)
Query parameter (α)	0.01, 0.1, **1**, 10, 100

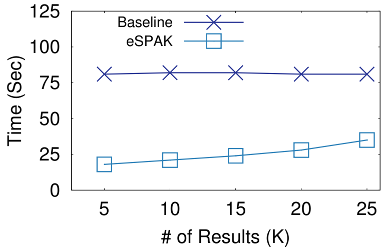

Figure 2: Effect of k on query processing time.

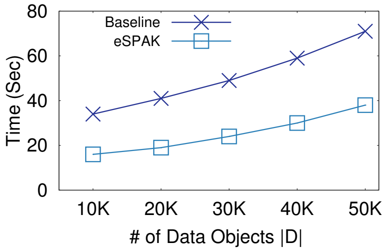

Figure 3: Effect of |D| on query processing time.

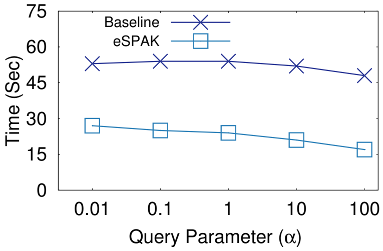

Figure 4: Effect of α on query processing time.

The description of each data object is extracted from twitter[1] messages, we assigned one tweet per data object. As a benchmark for eSPAK, we use a baseline method that computes the score of every data object using the incremental network expansion [9]. Both the algorithms are implemented in Java and executed on a desktop PC 2.80 GHz, Intel Core i5 with 8GB memory. In the experiments, we compared the query processing time of both algorithms. Table 1 summarizes the parameters used in the experiments. In each experiment, we vary a single parameter within the range that is shown in Table 1 while keeping the other parameters at the bolded default values.

5.2 Experimental Results

Figure 2 shows the query processing time for eSPAK and baseline as a function of the number k of requested data objects with the highest score. The query processing time of the baseline is nearly stable regardless of the k value because it always computes the score of each data object. However, the query processing time of eSPAK increases slightly with the k value. In Figure 3, we evaluate the performance of eSPAK and baseline by varying the cardinality of data objects. The query processing time of both the algorithms is sensitive towards an increase in |D|. However, eSPAK scales much better than baseline. Figure 4, demonstrates the impact of query parameter α on query processing time. A small value of α indicates more importance of textual relevance, whereas a high value of α gives more preference to the spatial relevance. Experimental results reveal that α does not indicate a significant impact on the query processing time of eSPAK and baseline. It is interesting to note that, both approaches performs better for higher values of α, which indicates more importance to spatial relevance. This is mainly because, when spatial relevance is higher, fewer edges are required to explore to find the top-k data objects.

6. CONCLUSIONS

In this paper, we investigate top-k spatial keyword queries in directed road networks. We presented an efficient indexing framework using inverted files, that indexes the data objects on edges which allows effective searching of data objects relevant to query in term of both textual and spatial relevance. Furthermore, we present an algorithm for evaluating top-k spatial keyword queries. Finally, the experimental evaluation conducted on real road networks demonstrates that eSPAK drastically reduces the query processing time compared to baseline algorithm.

7. ACKNOWLEDGMENTS

We thank anonymous reviewers for their valuable comments and suggestions. This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education(2016R1D1A1B03934129). Finally, this work was partially supported by Leaders Industry-University Cooperation Project.

8. REFERENCES

[1] Real datasets for spatial databases. http://www.cs.fsu.edu/~lifeifei/SpatialDataset.htm.

[2] H.-J. Cho, K. Ryu, and T.-S. Chung. An efficient algorithm for computing safe exit points of moving range queries in directed road networks. *Information Systems*, 41:1–19, 2014.

[3] G. Cong, C. S. Jensen, and D. Wu. Efficient retrieval of the top-k most relevant spatial web objects. *Proceedings of the VLDB Endowment*, 2(1):337–348, 2009.

[4] I. De Felipe, V. Hristidis, and N. Rishe. Keyword search on spatial databases. In *2008 IEEE 24th International Conference on Data Engineering*, pages 656–665. IEEE, 2008.

[5] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[6] L. Guo, J. Shao, H. H. Aung, and K.-L. Tan. Efficient continuous top-k spatial keyword queries on road networks. *GeoInformatica*, 19(1):29–60, 2015.

[7] R. Hariharan, B. Hore, C. Li, and S. Mehrotra. Processing spatial-keyword (sk) queries in geographic information retrieval (gir) systems. In *Scientific and Statistical Database Management, 2007. SSBDM'07. 19th International Conference on*, pages 16–16. IEEE, 2007.

[8] Z. Li, K. C. Lee, B. Zheng, W.-C. Lee, D. Lee, and X. Wang. Ir-tree: An efficient index for geographic document search. *IEEE Transactions on Knowledge and Data Engineering*, 23(4):585–599, 2011.

[9] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao. Query processing in spatial network databases. In *Proceedings of the 29th international conference on Very large data bases-Volume 29*, pages 802–813. VLDB Endowment, 2003.

[10] J. B. Rocha-Junior and K. Nørvåg. Top-k spatial keyword queries on road networks. In *Proceedings of the 15th international conference on extending database technology*, pages 168–179. ACM, 2012.

[1] http://twitter.com