User-Contributed Relevance and Nearest Neighbor Queries

Christodoulos Efstathiades[1] and Dieter Pfoser[2,3]

[1] Knowledge and Database Systems Laboratory
National Technical University of Athens, Greece
cefstathiades@dblab.ece.ntua.gr
[2] Research Center "Athena", Maroussi, Greece
pfoser@imis.athena-innovation.gr
[3] George Mason University, Fairfax, VA, USA
dpfoser@gmu.edu

Abstract. Novel Web technologies and resulting applications have lead to a participatory data ecosystem that when utilized properly will lead to more rewarding services. In this work, we investigate the case of Location-based Services and specifically of how to improve the typical location-based Point-Of-Interest (POI) request processed as a k-Nearest-Neighbor query. This work introduces Links-of-interest (LOI) between POIs as a means to increase the relevance and overall result quality of such queries. By analyzing user-contributed content in the form of travel blogs, we establish the overall popularity of a LOI, i.e., how frequently the respective POI pair is mentioned in the same context. Our contribution is a query processing method for so-called k-Relevant Nearest Neighbor (k-RNN) queries that considers spatial proximity in combination with LOI information to retrieve close-by and relevant (as judged by the crowd) POIs. Our method is based on intelligently combining indices for spatial data (a spatial grid) and for relevance data (a graph) during query processing. An experimental evaluation using real and synthetic data establishes that our approach efficiently solves the k-RNN problem when compared to existing methods.

1 Introduction

Location-based Services have been at the forefront of mobile computing as they provide an answer to the simple question as to what is around me. A lot of effort has been dedicated to improving such services typically by improving the selectivity of each request. Rating sites augment POIs with quality criteria. Preferences, when available, add further user specific parameters to a request. Context limits the available information based on situational choices. However, what has not been captured yet are user experiences per se, i.e., assessing what people want in terms of what people in the same situation have done in the past. Our objective is to provide location-based services, specifically relevant k-NN search based on the crowdsourced choices and experiences other users had in the past.

This work focusses on semantically enriching k-RNN search by taking user experience into account. Specifically, we introduce the concept of a Link-of-interest (LOI) between two POIs to express respective *relevance*, i.e., find related nearest POIs to my location. Relevance is inferred by extracting pairs of POIs that are frequently mentioned together in the same context by users. In our work, we extract this information (co-occurrence of POI pairs) by parsing travel blogs and using the page structure to derive relevance. This relevance information can be represented by means of a graph in which POIs represent nodes and LOIs are links. At the same time, we need to capture the spatial properties of POIs. The challenge in this work will be as to how we efficiently combine relevance captured by a graph and the spatial properties captured by a spatial index.

While existing work addresses spatio-textual search, i.e., introducing a spatial aspect to (Web) search, thus, enabling it to index and retrieve documents according to their geographic context, to the best of our knowledge, the exact problem of combining user experience (expressed as relevance) with spatial proximity in the form of k-RNN search (R stands for Relevance) has not been studied in literature. In this work, we are trying to solve the problem of how to combine searching a graph structure with a spatial index so as to efficiently process k-RNN queries. The spatial aspect is indexed by a static spatial grid, while the relevance is captured by a graph. We propose two methods, (i) GR-Sync (GR = Grid/Graph), which expands the two indexes separately, but synchronizes their search at certain steps and (ii) GR-Link, which uses a tighter integration in that spatial search results are seeded to the graph search so as to minimize costly (and most often unnecessary) expansions. Experimentation shows that the performance of GR-Link is best since it examines smaller portions of the data.

The outline of the remainder of this work is as follows. Section 2 discusses related work. Section 3 provides background on the data (spatial+relevance), queries (k-RNN), and the basic access methods that we use. The respective k-RNN query processing approaches are outlined in Section 4. Section 5 details the experimental evaluation and Section 6 concludes and provides directions for future research.

2 Related Work

To the best of our knowledge, the exact problem of combining user experience (expressed as relevance) with spatial proximity has not been studied in literature. Our methods are somewhat related to the research on Spatio-Textual Search. According to [16], studies in this area try to make web search geographically-aware, thus enabling it to index and retrieve documents according to their geographic context. Current research focusses on combining spatial and textual indexes and proposes hybrid methods to support geographical awareness. Current approaches are dealing with merging R-trees, regular grids, and space-filling curves with a textual index such as inverted files or signature files. One of the first works in Spatio-Textual indexing was conducted in the context of the SPIRIT search

engine [16]. SPIRIT facilitates the use of regular grids as spatial indexes and inverted files for the indexing of the documents. Following this idea, Zhou et al. [17] present hybrid indexing approaches comprising of R*-trees [1] for spatial indexing and inverted files for text indexing. Several approaches following the combination of R*-trees and inverted files followed [17, 4, 15]. Chen et al. [3] use space-filling curves for spatial indexing, whereas in [5], R-trees are combined with signature files in the internal nodes of the tree, so that a combined index is created. The algorithm in [9] is used for incremental nearest neighbor queries. Cong et al. [4] propose the IR-tree index, creating an inverted file for each node of an R-tree, again using [9]. Here, linear interpolation is used [13], in order for the spatial distance to be combined with the textual distance and therefore produce a combined score. In our work, we use a similar approach when combining the relevance score with spatial distance. A very related approach to [4] is presented in [11]. Recently, Rocha-Junior et al. [15] propose the hybrid index S2I, which uses aR-Trees [14]. The authors claim to outperform all other proposed approaches in this problem domain. Our work differs significantly in that we do not consider textual similarity, but use the co-occurrence of POIs in text as an indicator for relevance. This relevance information (and no actual textual information) is then combined with the spatial aspect of the POIs to retrieve relevant nearest neighbors.

To the best of our knowledge, the most related work that combines user experiences and spatial data is [2], which introduces the notion of prestige to denote the textual relevance between nearby to the query point objects. It also uses a graph, but the way it is constructed is based on the spatial distance between the objects as opposed to our method of using relevance. The prestige from a given query point is propagated through the graph, an information that is later used to extend the IR-tree [4]. Compared to our work, our index does not include a factor of randomness in our relevance score calculation as this is the case with PageRank-based algorithms. In general, the notion of relevance in [2] is based on different measures that are query dependent (textual relevance), a fact that is not being considered in our work. Our query considers only the location of a query point. Relevance is only derived from the collected dataset.

The problem of combining relevance and spatial distance is closely related to the problem of computing top-k queries that are based on different subsystems such as studied by Fagin et al. [7, 8]. The difference to our approach is that we have a specific focus on spatial data and on how to combine the result sets. Also, in our final k-RNN list we have the exact scores, compared to the approximate scores of the NRA algorithm.

Finally, in this work relevance is computed by counting the co-occurrences of POIs in the same paragraphs of texts. In spite of the simplicity of this metric, recent results show that POIs co-occur in documents when they are spatially close, have similar properties, or interact with each other [12]. Essentially, these observations are a confirmation of Tobler's first law of geography that states that "everything is related to everything else, but near things are more related than distant things."

3 Data and k-RNN Queries

With the proliferation of the Internet as the primary medium for data publishing and information exchange, we have seen an explosion in the amount of online content available on the Web. In addition to professionally-produced material being offered free on the Internet, the public has also been encouraged to make its content available online to everyone as User-Generated Content (UGC). We, in the following describe the data we want to utilize and how to exploit it to provide k-RNN queries.

3.1 Data

Web-based services and tools can provide means for users through attentional (e.g., geo-wikis, geocoding photos) or un-attentional efforts (e.g., routes from their daily commutes) to create vast amounts of data concerning the real world that contain significant amounts of information ("crowdsourcing"). The simplest possible means to generate content is by means of *text* when (micro) blogging. Any type of text content may contain geospatial data such as the mentioning of POIs, but also data characterizing the relationship between two POIs, e.g., the Red Cross hospital is next to the St. Basil church.

In this work, what we try to discover in these data sources are collections of POIs. Consider the example of Figure 1a. In the specific text snippet of a travel blog, three distinct spatial objects are mentioned, $O_1 = \{\text{Acropolis}\}$, $O_2 = \{\text{Plaka}\}$, and $O_3 = \{\text{Ancient Agora}\}$. We introduce the concept of *Link-of-Interest* (LOI) as a means to express relevance between two POIs. Assuming $O = \{O_1, \ldots, O_n\}$ is the set of all discovered POIs, then a text paragraph $P_x \subset O$, i.e., contains a set of POIs. It also holds that $O = \bigcup P$. We state that there exists a LOI $L_{i,j}$ between two POIs O_i and O_j, if both POIs are mentioned in the same text paragraph P_x. The set of all LOIs is defined as follows.

$$L = \{L(O_i, O_j) | \exists P_x \in P : O_i \in P_x \wedge O_i \in P_x\} \tag{1}$$

Do note that the definition of relevance is a very simple one, i.e., co-occurrence in the same paragraph. In future work, we intend to exploit the entire document structure as well as use sentiment information. However, any of these considerations will only affect the way we compute relevance and not the presented techniques for computing the specific type of query.

3.2 k-RNN Queries

Combining relevance information with spatial distance will allow us to provide better query results, i.e., POIs that are close-by and relevant.

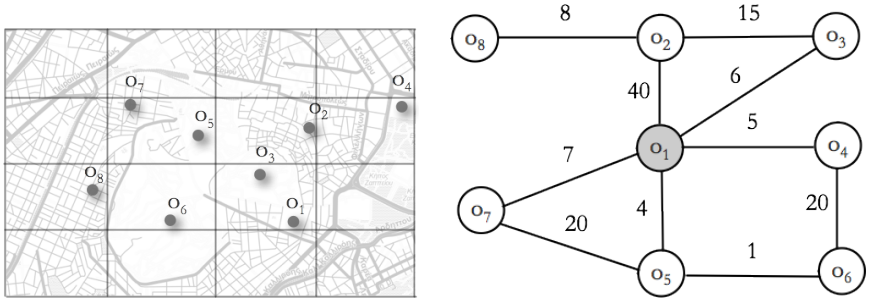
Let D be a spatial database that contains spatial objects O and Links of Interests L. Each spatial object is defined as $O_i = (O_i.id, O_i.loc)$ and each LOI as $L_k = (O_i.id, O_j.id, r)$, $O.id$ is a unique object identifier, $O.loc$ captures the object location in two-dimensional space, and r provides the relevance (score)

(a) POIs in text (b) POIs on a map

Fig. 1: "Closeby" Points Of Interest

of a LOI existing between two POIs o_i and o_j. The relevance r is computed as the number of times a specific POI pair appears in different paragraphs, i.e., the more often, the higher r and the higher the relevance of this specific LOI.

We introduce the k-RNN query as follows. Given a query point that is represented as a POI, find the k relevant nearest neighbors by taking into account, both, spatial proximity and relevance. An example is given in Figure 2 for a 1-RNN neighbor. Essentially, the query takes into account how relevant a close-by POI is to the query point and combining this information with the spatial proximity between the two POIs computes a combined k-RNN score denoting the Spatio-Relevance Distance between the two points, i.e., the smaller the distance (score), the better.



(a) Map (b) Relevance graph

Fig. 2: Relevance Graph and spatial "map" data

Equation 2 provides a means to compute a combined k-RNN score s_{rnn} that considers, both, spatial distance and relevance.

$$s_{rnn} = \frac{\alpha * s_r(Q,O)}{s_r.max} + \frac{(1-\alpha) * s_d(Q,O)}{s_d.max} \tag{2}$$

with $0 \leq \alpha \leq 1$ and where Q is the query point, O is the spatial object for which we compute the score with respect to Q, s_r is a score that takes into account the relevance score between two points and can be any kind of metric, $s_r.max$ is the maximum possible relevance score, s_d is the Euclidean distance between Q and O and $s_d.max$ is the maximum possible value the Spatial Distance can take and depends of course on the query space. Parameter α is used to denote the importance of each distance function (Relevance or Spatial) and can be tuned according to the user's needs.

In the next sections, detailed explanations are given as to the calculation of the Relevance Score and the computation of k-RNN queries.

3.3 Access Methods

Given a query and trying to define an efficient method for solving it in a data management context, typically access methods are used to speed up processing. In the following, we will discuss some methods that, either on their own, or, by combining them, efficiently solve the given query processing problem. When considering efficiency, we will also argue for the simplicity of a method, as the more complex a proposed access method is, the bigger is the challenge in implementing it in a given data management infrastructure.

Indexing Space and Relevance. Following this approach, we try to utilize spatial indexing methods with graph data structures. A regular spatial grid is used for indexing the locations of our POIs. The reason for using a regular grid instead of other types of access methods such as the R-tree is that (i) it can be updated in $O(1)$ time, (ii) there are no limitations to the index size (whole planet), (iii) k-NN queries can be processed incrementally by radiating out from the query point (see in the following), and (iv) it is a data structure that serves as a simple and elegant way of showing how a spatial index can be combined with others to index, e.g., space + relevance. The disadvantages of the method are that (a) space is divided into grid cells, which are considered to be disk pages and because of the static and non-uniform nature of the spatial data, pages will be underutilized and (b) the query time is not competitive when compared to other methods such as the R-tree and its more efficient variants.

However, in this work we do not consider the efficiency of our approach based merely on the use of the spatial index, but on how to create a hybrid method to answer k-RNN queries. Therefore, any type of spatial index can be used to replace the regular grid without having to change the algorithms used in our approach.

The *Relevance Graph* is defined as a graph $G(V, E)$, where V is the set of vertices that correspond to the POIs found in the set of documents, and E is the set of edges that correspond to links-of-interest (LOIs) between the POIs. The edge weights denote the relevance score r between a pair of vertices.

We consider a pair of POIs to be related if there is at least one co-occurrence in a paragraph and the relevance is derived from the number of co-occurrences.

For example, in Figure 2, we can see that the *Acropolis Metro Station* (O_1) and *Parthenon* (O_3) appear 6 distinct times in the same paragraphs, therefore their relevance score is 6.

Index Creation and Maintenance. Both indices are built incrementally during a pre-processing phase. In general, we consider that the data to be inserted concern one document/description at a time. Therefore, the input to the insert and update procedures are POIs and LOIs.

In the case of the Spatial Grid, considering its static nature, the corresponding grid cell is located based on the POI's coordinates and it is added to the respective node (corresponding to a cell). Since we aim for having a one-to-one correspondence between nodes and pages, should the node reach is maximum capacity, an overflow page is added.

For the Relevance Graph, we have to update the graph (add new points/vertices) and also update the relationships expressed as edge weights. Therefore, with the input of a list of points, as well as the identified LOIs, if a POI has not been previously added to the graph, we add it and also add the possible relations to other POIs. To perform this operation efficiently, a separate data structure is used to store the edges of the graph, so that they can be updated or added in $O(1)$ time.

4 *k*-RNN Query Processing

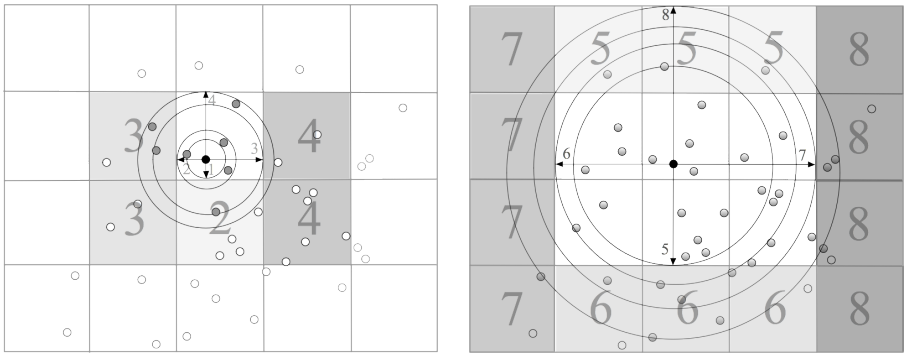
Having outlined structures to index the data, the major contribution of this work is how to integrate those methods so as to efficiently support the processing of k-RNN queries.

4.1 Index Synchronization

In order to tackle efficient k-RNN processing, we first define a basic query processing method, termed *GR-Sync* (derived from Grid/Graph synchronization) that combines the results of the two separate indices in order to answer k-RNN queries. GR-Sync consists of two separate methods for identifying the k-RNN candidates in the Spatial Grid and in the Relevance Graph, respectively. The intuition behind this approach is an intermixed, stepwise execution of the search in both indexes. After each step of the so-called *expansion process* in both data structures, the results (current list of respective k-NN neighbor candidates) are combined. If the neighbors found are guaranteed to be the k-RNN neighbors, then the procedure stops, otherwise it continues until the k-RNN neighbors are guaranteed to have been found.

Spatial Search. The *Spatial Expansion* algorithm uses a Spatial Grid that divides the surface of the earth into equal-sized grid cells as shown in Figure 2a. For each inserted POI we store four distances to the respective sides of the cell.

Figure 3 shows an example of how the algorithm behaves for eight expansions beginning from a query POI. The algorithm first locates the grid cell of the query point. Given that this algorithm tries to establish the relevance and proximity between POIs ("Where to go next?"), the query point is recruited from the set of POIs. The objective of the spatial expansion is to discover the nearest-neighbor



(a) Steps 1-4 (b) Steps 5-8

Fig. 3: Spatial expansion

POIs of the query point. We do so, by retrieving close-by POIs in neighboring cells in a step-wise fashion. As we will see, this results in *snail-like expansion* . The order in which the cells are retrieved depends on the location of the query point within its containing cell. Consider the example of Figure 3a. The query point is closest to the bottom side of the cell (arrow labeled "1"). To guarantee that all POIs have been examined that are within distance 1, only the cell of the query point needs to be loaded (indicated also by the circle around the query point and of radius "1"). However, for the case of distance "2" also the points of the bottom cell (labeled 2) need to be retrieved. This *snail-like expansion* next retrieves two cells labeled "3" and then two cells labeled "4". While we also retrieve points that are further away than d_4, we are also certain that we have not missed any candidates.

The points that are not within the maximum distance are added to a *retrievedPoints* list, whereas the points that are within the maximum distance are added to the k-NN list based on Euclidean distance.

With increasing distance, the search expands to neighboring cells as shown in the example of Figure 3b. As expected, the larger the distance from the query point, the more points are retrieved in each step, e.g., 4 cells in Step 8.

Computing the Relevance Score. To retrieve the *relevance score* s_r, we have to examine the Relevance Graph. Our method orients itself on the Breadth-First graph traversal. It starts with the query point and in a first step examines all

adjacent nodes in the Relevance Graph. Subsequent steps examine all neighbors of the initially-visited nodes and so on.

To compute a Relevance Score, we use the following recursive formula that computes the score of a node k based on the score of its predecessor, or, parent node p. For the initial expansion $p = q$.

$$s_r(k) = 1 - \frac{w_{p,k}}{\sum_{i=1}^{N} w_{p,i}} + s_r(p) + s_h(k) \tag{3}$$

$$s_h(k) = s_h(p) + h(k) \tag{4}$$

where s_r is the *Relevance Score* for node k, $w_{p,k}$ is the weight from the parent node p to k, N is the number of the parent's one-hop neighbors, the number of nodes that are expanded during the same step, $s_r(p)$ is the parent's relevance score, and $s_h(k)$ is a score derived from the number of hops needed to reach k from q. $s_h(k)$ is based on the respective score of the parent node and increases with the distance of k from d. This also ensures that any node l with $h(l) > h(k)$ will have a higher relevance score than node k, i.e., $s_r(l) > s_r(k) : h(l) > h(k)$. Observe that, both, the Relevance Score and also the Spatial Score are a penalizing score as they reflect distance, i.e., the higher s_r, the less relevant a node k with respect to q.

Figure 4 depicts an example of the calculation of the Relevance Score for the neighbors of a query point. First, the sum of the weights of the edges that connect the query point to its neighbors is calculated. In this example $\Sigma(0) = 28$. To compute each one-hop neighbor's Relevance Score s_r, we need to get the intermediate score of each neighbor node based on the edge weight, i.e., $1 - (w_{p,k}/\sum_{i=1}^{N} w_{p,i})$. For example, node B has an intermediate score of $1 - 3/28 = 0.89$. Applying the rest of Equation 3, with $s_r(p) = 0$, since $p = q$ and $h = 0$ with the hop count starting at 0, $s_r(B) = 0.89$. Continuing the expansion, i.e., retrieving the two-hop neighbors, for example for node F, $s_r(F) = (1 - 2/6) + 0.89 + 1 = 2.56$ (the edge that connects the neighbor to its parent is not taken into consideration). It should be noted that in the case of computing the score of a node, we consider the closest path as far as the number of hops from the query point is considered. Additionally, should a node be reachable by the same number of hops through multiple nodes, we consider the lowest scoring node as a parent node.

Synchronizing Expansion. To compute the k-RNN score, both, the spatial search and the relevance graph search need to be synchronized and their scores combined. The following approach computes combined scores and evaluates the status of the search at fixed intervals determined by the number of expansions performed in each index. As described earlier, "expansions" are performed to retrieve k-RNN candidates. In the Spatial Grid, this results in a snail-like expansion and retrieval of cells surrounding the query point, and in the Relevance Graph, a BFS-like expansion of increasing distance to the query point is used. After each expansion step, the two lists contain the closest in terms of spatial

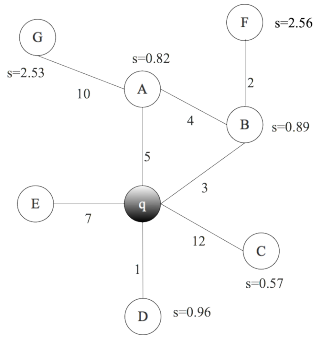


Fig. 4: Graph expansion and Relevance Score

and relevance score neighbors, respectively. The two scores in both lists need to be combined to assess k-RNN candidates. To synchronize the spatial expansion in the grid with the expansion of the Relevance Graph, we define a respective rate of expansion steps. The expansion ratio χ determines the ratio of spatial to relevance graph expansions. χ will typically be in the range of 4 to 16, as spatial expansions are considerably cheaper. Each search maintains a list of expanded POIs and their respective score. After each expansion cycle (considering the expansion ratio), the two lists are checked and the common POIs, i.e., appearing in both lists, are identified. Using Equation 2, their combined score s_{rnn} is computed. All POIs with such a score are added to a queue sorted by s_{rnn} essentially containing all top k-RNN neighbors that have been identified at this point, i.e., POIs that have been examined in both data structures.

To define a termination criterion for the search, we have to guarantee that all k-RNN POIs have been found, i.e., further search will not reveal any POIs with a better score than the ones already identified. Each search keeps an *open list* for each set of respective POIs found so far recording the corresponding k-RNN score. If a POI is found in only one index, to compute its score, we use a best-case estimate for the missing score, i.e., that it will be found during the next expansion. For example, assuming a POI was retrieved in the spatial search but not yet in the relevance search, the relevance score will be the lowest possible score after the next expansion (relevance score is dominated by number of hops = expansion steps). Similarly, should the spatial score be missing, we assume the best case, i.e., that the POI will be discovered during the next round of expansion with a distance from the query point just beyond the current search distance. As the searches progress, the scores of the POIs, and thus the open lists, will be updated based on current number of hops and search distance. The GR-Sync algorithm is shown in Figure 5. GR-Sync uses s^d_{rnn} and s^r_{rnn} as the minimum predicted combined k-RNN scores for nodes with no valid spatial, or relevance score, respectively. If $s^*_{rnn} = min(s^d_{rnn}, s^r_{rnn})$, the minimum score that still could be found, is less than $max(s_{rnn})$, i.e., the maximum score of identified rnn candidates, it means that the POIs in the (top-k) result list are

guaranteed to be the top k-RNN neighbors (Line 17). This condition can be used as termination criterion.

GR-SYNC(q, k)

1 RG ▷ Relevance Graph
2 SG ▷ Spatial Grid
3 rg ▷ Discovered POIs in RG
4 sg ▷ Discovered POIs in SG
5 rnn ▷ k-RNN result list
6 n ▷ Relevance Graph step count
7 χ ▷ Expansion ratio Spatial Grid
8 **while** ¬ complete
9 SPATIALEXPANSION$(q, k, (n \times \chi), SG, sg)$
10 SEMANTICEXPANSION(q, k, n, RG, rg)
11 $rnn = sg \cap rg$ ▷ NN results with complete score
12 s^r_{rnn} = MINSCORE(rg) ▷ Min. predicted score, Relevance Graph
13 s^d_{rnn} = MINSCORE(sg) ▷ Min. predicted score, Spatial Grid
14 s'_{rnn} = MIN$(s^d_r nn, s^r_r nn)$ ▷ Min. expected score
15 s^*_r = MAXSCORE(rnn) ▷ Max. score in current result list
16 ▷ if the max k^{th} RNN distance found so far is less than the min predicted distance
17 complete = $(|rnn| \geq k \wedge s^*_r < s'_r nn)$

Fig. 5: GR-Sync k-RNN algorithm

4.2 Index Linking

The k-RNN query processing algorithm presented so far does not actually combine the two indices (spatial and relevance) in any way, but only evaluates the results at times with the expansion steps used as a means of synchronization. An inherent problem with this method is the search in the Relevance Graph, which after the first hops becomes very costly. Big expansions in the Relevance Graph (empirically observed for hops > 3) retrieve a lot of data and, hence, incur disk activity. To address this problem, we have devised a variation of our query processing technique that manages to bound the expansion in the Relevance Graph, i.e., to compute the Relevance Scores without expanding large potions of the graph. In this *GR-Link* method, termed as such since it combines the two indexes (Grid/Graph linking), we use the results of the spatial search as seed elements to the search in the Relevance Graph. The intuition is that many 1-hop expansions (spatial search seeds) are "cheaper" than a single n-hop expansion of the query point. POIs retrieved by the spatial search (and not discovered yet in the relevance search) are expanded in the Relevance Graph (cf. Figure 6) The intuition is that the graph searches of (i) the seeded POIs and (ii) the query

Fig. 6: GR-Link method using Grid POIs in Graph search

GR-LINK(q, k)

1 RG ▷ Relevance Graph
2 SG ▷ Spatial Grid
3 rg ▷ Discovered POIs in RG
4 rg' ▷ Discovered POIs in RG - spatial seeds
5 sg ▷ Discovered POIs in SG
6 rnn ▷ k-RNN result list
7 n ▷ Relevance Graph step count
8 χ ▷ Expansion ratio Spatial Grid
9 **while** ¬ complete
10 SPATIALEXPANSION$(q, k, (n \times \chi), SG, sg)$
11 **for** each $poi \in sg$
12 SEMANTICEXPANSION$(poi, 1, RG, rg')$
13 SEMANTICEXPANSION(q, n, RG, rg)
14 $rg =$ CONNECT(rg, rg') ▷ Combine graph expansions
15 $rnn = sg \cap rg$ ▷ NN results with complete score
16 $s^r_{rnn} =$ MINSCORE(rg) ▷ Min. predicted score, Relevance Graph
17 $s^d_{rnn} =$ MINSCORE(sg) ▷ Min. predicted score, Spatial Grid
18 $s'_{rnn} =$ MIN$(s^d_r nn, s^r_r nn)$ ▷ Min. expected score
19 $s^*_r =$ MAXSCORE(rnn) ▷ Max. score in current result list
20 ▷ if max k^{th} RNN distance found so far is less than min predicted distance
21 complete $= (|rnn| \geq k \wedge s^*_r < s'_r nn)$

Fig. 7: GR-Link k-RNN algorithm

point will meet eventually and, thus, we can compute a POI's relevance score. Without *seeding the search*, we would have to wait until the query point expansion reaches a POI. The simplified example of Figure 6 should illustrate that the POIs expanded are much fewer than the POIs that would have been retrieved if a second expansion step from the query point would have been performed.

The GR-Link approach also allows us to better bound the estimates of missing scores. Placing a POI on the Relevance Graph, we know for sure that if their expansions (POI and query point) do not meet, the base for their score computation is at least the sum of the two expansions plus one hop (since they did not meet). Therefore, the predicted s_r score for such a point is larger than what it would have been in the non-hybrid case.

The pseudo code for this method is shown in Figure 7. The difference from the GR-Sync algorithm are the statements in Lines 11-14, which seed POIs to the Relevance Search.

As the experimental section will show, the intuition of choosing many small Relevance Graph expansions over one large expansion pays off and the GR-Link

method shows superior performance in terms IO when compared to the GR-Sync solution.

5 Experimental Evaluation

Did the previous sections define our approach to the processing of k-RNN queries, so will we in the following establish its efficiency. An empirical study using real and synthetic datasets will assess the performance of the query processing methods in terms of accessed data (disk I/O operations) under varying parameter settings. Overall, we compare three methods, (i) the GR-Sync method (naive method), (ii) the GR-Link method and (iii) an hypothetic ideal method (see below for an explanation). What we expect from the experiments is to see a well-performing GR-Link method that comes close to the performance of the ideal method.

5.1 Data

The experimentation relies on real and synthetic datasets. The real data consists 120k POIs and 670k LOIs extracted from a corpus of 120k documents. The documents were collected from three different travelblog sites (travelblog, traveljournals, travelpod) as described in [6]. The texts were pre-processed to collect the necessary information for the index: (i) identification of POIs, (ii) geocoding of POIs (spatial position), and (iii) location within the document (paragraph id and offset).

To improve the significance of the experiments, we also generate a larger-in-size synthetic dataset. The procedure for generating the data follows simple heuristic rules derived from the characteristics of the real data, i.e., generating hotspots (cities) with POIs and links between these hotspots. We generate center points (cities, attractors) in an area of 30×30 degrees (approximately 3300 $\times$ 3300km) using a uniform distribution. For each center point, using a normal (gaussian) distribution, we generate neighboring points (POIs). We then generate relationships (LOIs) between POIs using a normal distribution, i.e., the neighbors that are spatially closer to a POI in question are more likely to be linked to it than the ones further away (Tobler's spatial bias). On average, we generate 30 LOIs per POI (maximum = 50). The edge weights (denoting the number of common paragraphs) are generated randomly based on observations from our real dataset and vary between 0 and 500. Our synthetic dataset consists of 1 million POIs that form a Relevance Graph with a total of 6.5 million edges (LOIs). It is considerably larger than the Relevance Graph derived from the real data and, thus, should allow us to provide a more conclusive experimental evaluation.

5.2 Experimental Setup

We evaluate the various query processing methods not only with different datasets but also a varying set of parameters including (i) the number of RNN neighbors

k and (ii) the preference parameter α emphasizing either spatial distance or relevance.

The performance is assessed in terms of number of page accesses (I/O) performed to retrieve the data from disk for each query. In each case, we performed 100 queries and computed the average I/O shown in the respective charts. An important aspect is the spatial grid used to index the data. We use a regular grid with a spacing of 0.02 degrees (approximately 2km) in longitude and latitude. The synthetic dataset consists of 1 million POIS covering an area of 30 degrees longitude and latitude, respectively, an extent somewhat comparable to Europe. The 1M POIs are grouped into 160k cells amounting to an average space utilization of 6, but not exceeding 30. Keep in mind that we only consider occupied cells in our index. The real dataset contains 120k points that are scattered all over the globe. Here, 80k cells contain at least one POI. This amounts to an average space utilization of < 2, with few cells containing more than 5 POIs. Essentially, we used this dataset as a template for the synthetic data generation. Still, we wanted to present the results of the performance study, to see whether the trends with respect to the indexing methods persist.

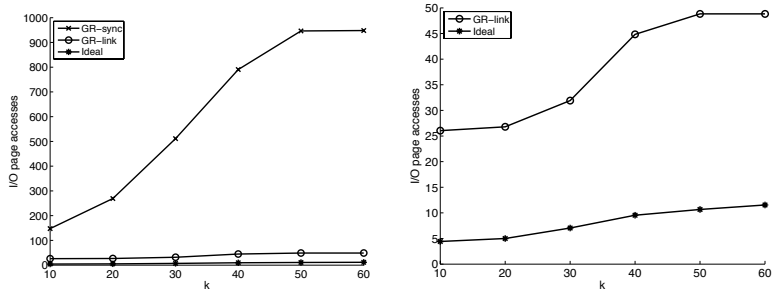
The expansion ratio χ was set to 12, i.e., for each expansion in the Relevance Graph, 12 expansions in the Spatial Grid are performed.

5.3 k-RNN Query Performance

In our experimental setup, we first vary the number of sought nearest neighbors k to see how scalable our algorithms are in terms of I/O. In this experiment, the parameter α is fixed to 0.5, i.e., considering the spatial and relevance score equally important.

As we can see in Figure 8a, the GR-Link approach outperforms the GR-Sync method by an order of magnitude. This is due to the fact that GR-Sync needs to search large parts of, both, the Spatial Grid and the Relevance Graph in order to guarantee the result. Therefore, when POIs are found in one of the two indices, it needs to keep searching the other to find the combined k-RNN score. This causes the algorithm to access a lot of unnecessary pages. On the other hand, the GR-Link method uses the results of the Spatial Search to limit the expansions in the Relevance Graph. Hence, this method has a considerably better performance.

While a comparison to the naive GR-Sync approach does not pose a challenge, so does the next experiment relate the performance of GR-Link to an "ideal" approach (cf. Figure 8b). The ideal approach simulates a Relevance Graph search that terminates as soon as the k-RNN POIs are found, i.e., we have a-priori knowledge of the results and are only expanding the graph until "discovered". As such this method is unrealistic, but should just serve as a lower-bound for the performance of the hybrid index. Figure 8b shows that the GR-Link method examines more data than the ideal approach (albeit little data overall). Still, in terms of comparison to the baseline approach, the hybrid index's performance is close to that of the ideal method.



(a) GR-Sync vs. GR-Link vs. Ideal (b) GR-Link vs. Ideal

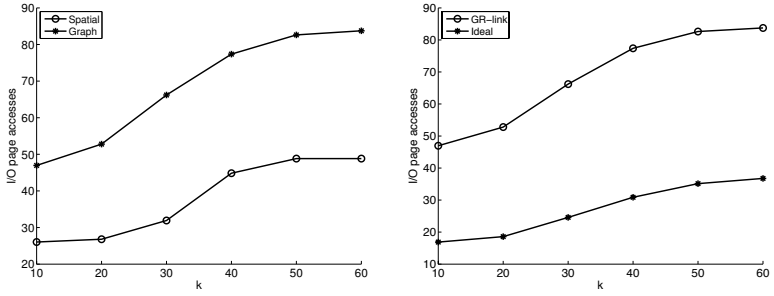
Fig. 8: Naive, Hybrid and Ideal Index I/O performance

5.4 Spatial Grid vs. Graph Partitioning

All POI data is kept on disk. The assumption so far is that each populated Spatial Grid cell constitutes a page on disk. This approach has the disadvantage of taking into consideration only the spatial characteristics without taking into account that the nearest-neighbor search explores not only the spatial grid but the Relevance Graph as well. The more the points discovered in the Relevance Search (graph expansion) are spatially spread out, the more pages need to be retrieved, thus, increasing the I/O cost. Our approach in extending the current disk layout is now to group POIs together based on the proximity in the Relevance Graph. This is achieved by partitioning the Relevance Graph using the METIS graph partitioning tool [10]. However, the grid-based partitioning is essential for spatial search and the underlying expansion mechanism. Using (relevance) graph-based partitioning, we mapped the spatial grid cells to graph partitioning. I.e., does the spatial search request a specific cell, the respective one or more graph partitions are fetched. Figure 9a shows that graph-partitioning does not provide an advantage over the static spatial partitions. What is interesting to see is that the performance advantage of the ideal method when compared to GR-Link is diminished for the case of graph partitioning. An explanation here could be that the expansion solely relies on the graph and, hence, a respective partitioning would provide some (respective) advantage for this method. Overall however, the ideal method performs best in the case of the Spatial Grid, which is evident when comparing Figures 8b and 9b.

5.5 Score Balance

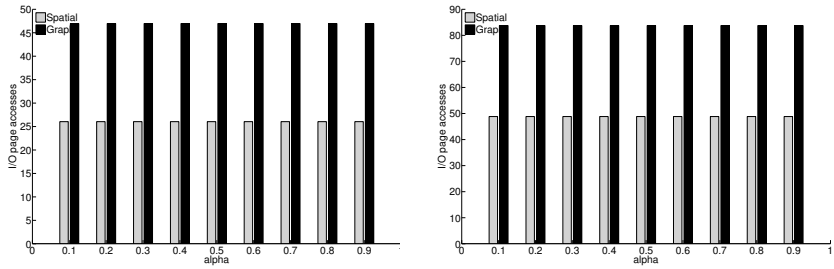
The preference parameter α balancing the effect of spatial distance vs. relevance on the query result is not only a critical factor for the quality of the result, but also affects the query processing cost. With $\alpha = 0.1$ it favors the Spatial Score and with $\alpha = 0.9$ it favors the Relevance Score. The two charts of Figure 10 show that the query cost does not change significantly with α. It seems that with an emphasis on Relevance ($\alpha > 0.5$), the cost slightly decreases due to fewer spatial expansions.



(a) GR-Link: Spatial Grid vs. Graph Partitioning

(b) Graph Partitioning: GR-Link vs. Ideal

Fig. 9: Partitioning: Spatial Grid vs. Relevance Graph



(a) k=10

(b) k=60

Fig. 10: Varying preference parameter α

5.6 Real Dataset Experiments

The number of page accesses when compared to the experiments with the synthetic dataset appear to be orders of magnitude greater in number because of the many spatial expansions that need to be performed in order for the algorithms to guarantee the k-RNN result in a sparse dataset. Overall, this experiment shows the same trends observed for synthetic data.

5.7 Summary

The experiments showed that one can provide an efficient indexing mechanism and query processing method for combining spatial and graph-based search by interlinking index traversal in both "spaces". The GR-Link method performs very close to an "ideal" method and an order of magnitude better than the naive GR-Sync method.

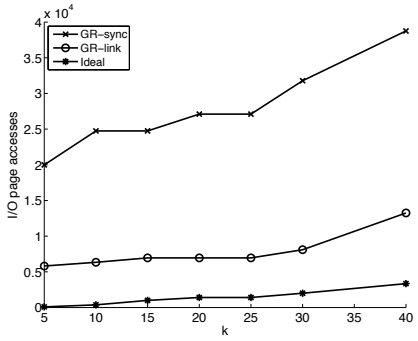


Fig. 11: Real Dataset: varying k

6 Conclusions

The motivation for this work was to find an efficient method to process k-RNN queries, i.e., a version of the NN problem that also considers the relevance between query points. Relevance in our case is defined as co-occurrence of POIs in texts (Links of Interests - LOI). While a rather simplistic measure, it is adequate to define and evaluate the proposed approach. To solve the k-RNN problem, we define two query processing methods that rely on a spatial grid and a graph to capture the respective data aspect. Experimentation shows that GR-Link is an efficient method that performs significantly better than a naive method and comes close to the performance of a hypothetical ideal method. This work outlines a first approach to combining spatial and graph search and consequently directions for future work are plenty. They include using more efficient spatial access methods, optimizing the search in the graph, and adding graph information to the spatial index. We aim also in performing an "interestingness" evaluation in order to measure the usefulness of the k-RNNquery to the users.

Acknowledgements

The research leading to these results has received funding from the European Union Seventh Framework Programme - Marie Curie Actions, Initial Training Network GEOCROWD (http://www.geocrowd.eu) under grant agreement No. FP7-PEOPLE-2010-ITN-264994.

References

1. N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The r*-tree: an efficient and robust access method for points and rectangles. In *Proc. SIGMOD conf.*, pages 322–331, 1990.
2. X. Cao, G. Cong, and C. S. Jensen. Retrieving top-k prestige-based relevant spatial web objects. *PVLDB*, 3(1-2):373–384, 2010.

3. Y.-Y. Chen, T. Suel, and A. Markowetz. Efficient query processing in geographic web search engines. In *Proc. SIGMOD conf.*, pages 277–288, 2006.
4. 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.
5. I. De Felipe, V. Hristidis, and N. Rishe. Keyword search on spatial databases. In *Proc. 24th ICDE conf.*, pages 656–665, 2008.
6. E. Drymonas and D. Pfoser. Geospatial route extraction from texts. In *Proc. 1st Workshop on Data Mining for Geoinformatics*, pages 29–37, 2010.
7. R. Fagin. Combining fuzzy information from multiple systems. *J. Comput. Syst. Sci.*, 58(1):83–99, 1999.
8. R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. *J. Comput. Syst. Sci.*, 66(4):614–656, 2003.
9. G. R. Hjaltason and H. Samet. Distance browsing in spatial databases. *ACM Trans. on Database Syst.*, 24(2):265–318, 1999.
10. G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20:359–392, 1998.
11. Z. Li, K. C. K. Lee, B. Zheng, W.-C. Lee, D. Lee, and X. Wang. IR-tree: An efficient index for geographic document search. *IEEE Trans. on Knowl. and Data Eng.*, 23(4):585–599, 2011.
12. Y. Liu, F. Wang, C. Kang, Y. Gao, and Y. Lu. Analyzing relatedness by toponym co-occurrences on web pages. *Transactions in GIS*, to appear, 2013.
13. B. Martins, M. J. Silva, and L. Andrade. Indexing and ranking in geo-ir systems. In *Proc. Geographic Information Retrieval Workshop*, pages 31–34, 2005.
14. D. Papadias, P. Kalnis, J. Zhang, and Y. Tao. Efficient olap operations in spatial data warehouses. In *Proc. 7th SSTD symp.*, pages 443–459. Springer-Verlag, 2001.
15. J. a. B. Rocha-Junior, O. Gkorgkas, S. Jonassen, and K. Nørvåg. Efficient processing of top-k spatial keyword queries. In *Proc. 12th SSTD symp.*, pages 205–222, 2011.
16. S. Vaid, C. B. Jones, H. Joho, and M. S. Spatio-textual indexing for geographical search on the web. In *Proc. 5th SSTD symp.*, pages 218–235, 2005.
17. Y. Zhou, X. Xie, C. Wang, Y. Gong, and W.-Y. Ma. Hybrid index structures for location-based web search. In *Proc. 14th CIKM conf.*, pages 155–162, 2005.