Top-k Spatial Keyword Preference Query

João Paulo Dias de Almeida and João B. Rocha-Junior

State University of Feira de Santana, Brazil
jpalmeida.uefs@gmail.com and joao@uefs.br

Abstract. With the popularity of devices that are able to annotate data with spatial information (latitude and longitude), more and more spatial data are becoming available. This has attracted the attention from the research community in the processing of advanced spatial queries. In this article, we study a new query type named Top-k Spatial Keyword Preference Query that selects objects of interest based on the textual relevance of other spatio-textual objects in their spatial neighborhood. This article introduces this new query type, presents three algorithms for processing the query efficiently and goes over an extensive experimental evaluation to study the performance of the algorithms proposed, employing real datasets.

Categories and Subject Descriptors: H.2.8 [**Database Applications**]: Spatial databases and GIS; H.2.4 [**Systems**]: Textual databases; H.2.4 [**Systems**]: Query processing

Keywords: Hybrid Access Methods, Preference Queries, Spatial Databases, Query Processing, Textual Databases

1. INTRODUCTION

With the popularization of GPS enabled devices, the volume of spatial data produced has increased significantly in the last years, which explains the interest for new approaches to extract relevant information from this amount of data [Cao et al. 2012]. Facebook, Google Maps, Twitter and Waze are examples of applications that process spatial data in order to obtain relevant information. Most of the spatial data available is associated with a text. For example, some messages of Twitter (text) sent from Smartphones have the spatial coordinates (latitude and longitude). The objects in the OpenStreetMap (www.osm.org) have a spatial location and are associated with a descriptive text. The objects that have spatial and textual information are so called *spatio-textual objects* [Vaid et al. 2005].

A significant part of the traditional spatial types of query is user centered. Most types of query search for spatial objects considering the user position. This is the case of the spatial queries *range* and *nearest neighbor* (*nn*). The range selects objects that are within a distance r (radius) of the user location, while *nn* returns the closest spatial object from the user location. This is also the case of the Top-k Spatial Keyword Query [Cong et al. 2009] that returns the k most relevant spatio-textual objects by considering both the distance between the spatio-textual objects and the user location, and the relevance between the text of the spatio-textual objects and the query keywords.

In this article, we propose a new query type named Top-k Spatial Keyword Preference Query. Differently from the user centered spatial types of query, this new query type searches for spatial objects of interest considering other spatio-textual objects in their spatial neighborhood. Specifically, given a set of spatial objects of interest (*e.g.* hotels), a set of spatio-textual objects of reference (*e.g.* bars, restaurants and tourist attractions), a spatial selection criteria (*e.g.* 100m from the spatial objects of interest) and a set of query keywords (*e.g.* "Italian food"); the Top-k Spatial Keyword

This work was funded by FAPESB and CAPES.

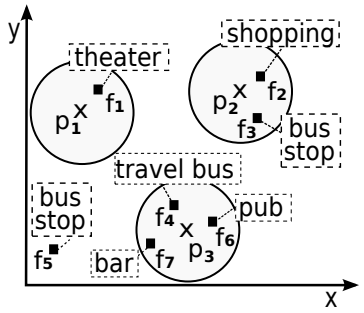


Fig. 1. Objects of interest (p) and objects of reference (f) associated with a text.

Preference Query returns the k best spatial objects of interest, where the score of each object is given by the highest textual relevance between the query keywords and the text of the spatio-textual objects of reference that satisfies the spatial selection criteria.

For example, Fig. 1 presents a spatial area (*e.g.* a city) with spatial objects of interest p (*e.g.* apartments for renting) and spatio-textual objects of reference f (*e.g.* any establishment). Thus, a user interested in renting an apartment close to a bus stop specifies the query keywords "bus stop" and defines the spatial selection criteria (the circle around the spatial objects of interest). The Top-k Spatial Keyword Preference Query returns the objects p_2 as top-1 and p_3 as top-2. The object p_2 is more relevant than p_3 because it has in its spatial vicinity the spatio-textual object of reference f_3 that is more textually relevant to the query keywords than any other spatio-textual object of reference in the vicinity of p_3. The object p_3 is more relevant than p_1. In the spatial vicinity of p_3, the object f_4 is textually relevant to the query keywords, while there is no textually relevant object in the vicinity of p_1. Note that f_5 is textually relevant to the query keywords, but it is not in the spatial vicinity of any spatial object of interest.

Several applications can benefit from this new query type in order to provide advanced location-based services. For example, an application for helping tourists can employ this query type for selecting the best metro stations according to the tourist interest. In this case, the tourist would select the metro stations to be the objects of interest, defining his preference through the query keywords (*e.g.* "open air museum"). This query type would return the k best metro stations (spatial objects of interest) near establishments (spatio-textual objects of reference) relevant to the query keywords. Government applications (e-Gov) can also benefit from this query type. For example, given a set of streets (spatial objects of interest) and tweets with spatial location (spatio-textual objects of reference), the query returns the streets that are more relevant to the query keywords "theft, robbery and shooting", indicating streets that should receive better attention from the police.

To the best of our knowledge, there is no other spatial or textual query type that is able to select the k best objects of interest, employing only the textual relevancy of the spatio-textual objects of reference in the spatial vicinity of the objects of interest. The two most related work are proposed by Yiu et al. [2007] and Tsatsanifos and Vlachou [2015]. The first does not consider the textual relevance of the objects of reference and the second requires a predefined score in the objects of reference, which is not available in most real datasets.

The main contributions of this article are:

—To propose an new query type named Top-k Spatial Keyword Preference Query;

—To present algorithms to process this query type efficiently with two different spatial selection criteria: range and nearest neighbor;

—To perform an experimental evaluation to study the algorithms proposed using real datasets.

The rest of this article is organized as follows: Section 2 presents the related work; Section 3 contains a precise definition of this new query type; Section 4 describes the algorithms to process this query type for range and nearest neighbor; Section 5 contains the experimental evaluation and, finally, Section 6 contains the final remarks.

2. RELATED WORK

The research in the field of spatio-textual queries can be separated in three main periods. In the first period, the research focus was on improving the results of the search engines, with an information retrieval perspective. The queries had the main purpose of retrieving relevant documents for the query keywords. Instead of searching for objects with a specific spatial location, the approaches employed the name of the places such as cites and countries, in order to search for relevant results [Zhou et al. 2005]. For example, in order to answer the query "Eiffel Tower in Paris", the keyword "Paris" is employed to define the place of interest, filtering out documents that are not relevant for this place.

In the second period, with the popularity of GPS enabled devices, the research focus was on identifying spatial objects relevant for the query keywords. The research did not focus in computing the precise textual relevance between the query keywords and the text of the spatial objects. Instead, they employed a Boolean approach by returning the objects that match the query keyword [De Felipe et al. 2008]. For example, "return all objects that are from 100m of the given query location (latitude, longitude) that have the keywords 'Eiffel' *and* 'Tower' in their description".

In the third period (current), the researchers are interested in the precise location of the spatial-textual objects and in the textual relevance between the query keywords and the text of the objects. The textual relevance is computed by employing text similarity approaches such as cosine [Zobel and Moffat 2006]. In order to compute the score of the objects, considering the precise spatial location and the textual relevance, new hybrid indexes have been proposed [Chen et al. 2013; Cong et al. 2009; Rocha-Junior et al. 2011].

These hybrid indexes combine spatial and textual indexes such as R*-tree [Beckmann et al. 1990] and Inverted Files [Zobel and Moffat 2006]. These indexes can be divided in two categories: spatial first or textual first. The spatial first indexes [Cong et al. 2009] include inverted files inside spatial indexes such as R*-tree. On the other hand, the textual first indexes include spatial indexes such as R*-tree in the Inverted Files. The textual first approach has presented better results in most cases and are employed in this work [Chen et al. 2013].

One of the most studied types of query in the third period is the Top-k Spatial Keyword Query [Cong et al. 2009; Rocha-Junior et al. 2011]. Given a query location (latitude and longitude) and a set of query keywords, this query returns the k best spatio-textual objects considering the distance between the objects and the query location, and the textual relevance between the objects and the query keywords. There is an equation[1] that combines the spatial distance and textual relevance, enabling to compute the score of each object precisely. The k objects with highest scores are returned. The Top-k Spatial Keyword Preference Query proposed in this article is different from the Top-k Spatial Keyword Query. Instead of searching for spatio-textual objects near a given location, it assumes a predefined set of objects of interest and searches for the top-k best objects of interest based on the text of other spatio-textual objects in their spatial neighborhood.

Another type of query of interest is the Traditional Top-k Spatial Preference Query [Yiu et al. 2007], which is described as follows. Given a set of spatial objects of interest P, a set of spatial objects of reference F_i, where each $f \in F_i$ have a predefined score, and a spatial neighborhood criteria; the Traditional Top-k Spatial Preference Query returns the k best objects $p \in P$, where the score of each

[1]The equation is $\alpha \cdot \theta + (1 - \alpha) \cdot \lambda$, where θ is the textual relevance and λ is the spatial distance [Rocha-Junior et al. 2011].

object p is the highest score among the objects of reference f that satisfy the spatial neighborhood criteria.

For example, Fig. 2 presents a spatial area containing spatial objects of interest p (*e.g.* hotels) and spatial objects of reference $a \in F_1$ (*e.g.* cafes) and $b \in F_2$ (*e.g.* bars). Each object of reference is associated with a predefined score. The circle around each spatial object of interest p represents the spatial neighborhood criteria defined by the user (*e.g.* 100m from the objects of interest). A user interested in a hotel near a good bar and a good restaurant will have p_1 as top-1 and p_2 as top-2, because the object of reference a_1 in the spatial neighborhood of p_1 has a higher score than a_3.

In the Traditional Top-k Spatial Preference Query, the score of the spatial objects of reference is known in advance, which allows materializing partial results to process this query efficiently [Rocha-Junior et al. 2010]. In the Top-k Spatial Keyword Preference Query proposed in this article, the spatio-textual objects of reference do not have a predefined score, but a text. Thus, in order to compute the score of the spatial objects of interest, the textual relevance between the query keywords and the text of the spatio-textual objects of reference must be computed. Processing the Top-k Spatial Keyword Preference Query is more challenging.

Recently, Tsatsanifos and Vlachou [2015] presented a type of query similar to the one proposed in this article. However, they assume that each spatio-textual object have a predefined score and a text, not only a text. This assumption is not practical for most datasets available in the Internet. Most datasets have a text and a spatial location and do not have a pre-defined score (ex. Twitter and Open Street Map). Moreover, the predefined score is employed in the proposed algorithms in order to improve the processing performance of this type of query. In our work, we assume that the spatio-textual objects of reference have only the text and we do not benefit from any predefined score in order to process this type of query.

3. SPECIFICATION OF THE TOP-K SPATIAL KEYWORD PREFERENCE QUERY

In this section, we specify the Top-k Spatial Keyword Preference Query proposed in this article. First, we present the datasets required for the query processing; next, we present the query parameters; and finally, we present the spatial selection criteria.

Given a set of spatial objects of interest P, where each object $p \in P$ has a spatial coordinate $p = (p.x, p.y)$; and a set of spatio-textual objects of reference $f \in F$, where each f has a spatial coordinate $(f.x, f.y)$ and a text $f.D$, $f = \{(f.x, f.y), f.D\}$. The Top-k Spatial Keyword Preference Query Q has three parameters $Q = \{Q.D, Q.\psi, Q.k\}$, where $Q.D$ is the set of query keywords, $Q.\psi$ is the spatial selection criteria, and $Q.k$ is the number of expected results.

The Query Q returns the $Q.k$ objects in P with the highest scores. The score of an object p, $\tau^{Q.\psi}(p)$, is the highest textual relevance (textual similarity) among all spatio-textual objects of reference f

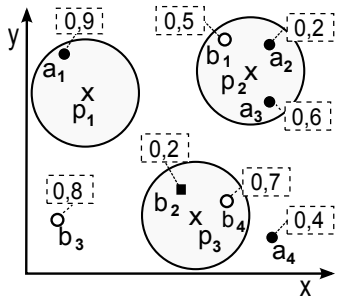


Fig. 2. Objects of interest (p) and objects of reference (a and b) with a score.

that satisfy the spatial selection criteria $Q.\psi$. The spatial selection criteria can be range ($Q.\psi = rng$), nearest neighbor ($Q.\psi = nn$) and influence ($Q.\psi = inf$) [Yiu et al. 2007].

—Given a radius r, the score of p assuming the range as the spatial selection criteria is:

$$\tau^{rng}\ (p) = max\ \{\ \theta(f.D, Q.D) \mid f \in F : dist(p, f) <= r\ \}$$

—The score of p assuming the nearest neighbor as the spatial selection criteria is:

$$\tau^{nn}\ (p) = max\ \{\ \theta(f.D, Q.D) \mid f \in F, \theta(f.D, Q.D) > 0, \forall\ v \in F : dist(p, f) \leq dist(p, v)\ \}$$

—Given a radius r, the score of p assuming the influence as the spatial selection criteria is:

$$\tau^{inf}\ (p) = max\ \{\ \theta(f.D, Q.D) \cdot 2^{\ -dist(p,f)/r} \mid f \in F\ \}$$

where $\theta(f.D, Q.D)$ is the textual relevance (textual similarity) between the text of the spatio-textual object of reference $f.D$ and the query keywords $Q.D$. In this article, we employ the cosine to compute the textual relevance as defined by Rocha-Junior et al. [2011] and the Euclidean distance $dist(p, f)$ between an object p and a spatio-textual object of reference f.

For example, in Fig. 1, the objects of interest P (apartments) are $\{p_1, p_2, p_3\}$, while the spatio-textual objects of reference F are $\{f_1, f_2, f_3, f_4, f_5, f_6, f_7\}$. The parameters of the query Q are $Q.D$ ="bus stop", $Q.\psi$ $=rng$ (with a radius of 200m), and $Q.k = 1$, which will return the best apartment for the query keywords "bus stop" taking into account the highest textual relevance among the objects f which are from a given distance (radius) of each object p.

With the range as the spatial selection criteria ($Q.\psi = rng$), the score of a spatial object of interest p is defined by the highest textual relevance $\theta(f.D, Q.D)$ among the spatio-textual objects of reference f whose the distance to p is smaller or equal the radius r, $dist(p, f) \leq r$. With the nearest neighbor ($Q.\psi = nn$) as the spatial selection criteria, the score of p is defined by the textual relevance $\theta(f.D, Q.D)$ of the nearest object f, if $\theta(f.D, Q.D) > 0$. If there is more than one textual relevant object f with the same smallest distance to p, the score of p is the highest textual relevance among the objects with the same smallest distance. Finally, with the influence as the spatial selection criteria ($Q.\psi = inf$), the score of p is defined by an equation that combines textual relevance and spatial distance. The larger the distance, the smaller the value returned by the equation. Among all objects f, the score of p is the highest value returned by the equation.

In this article, we present algorithms to process the Top-k Spatial Keyword Preference Query, employing the range and nearest neighbor as spatial selection criteria.

4. PROPOSED APPROACHES

In this section, we present the algorithms proposed to process the Top-k Spatial Keyword Preference Query. The first algorithm is the base for the other two algorithms proposed. The first algorithm IFA (Inverted File based Algorithm), employs an Adapted Inverted File to store the set of objects of reference. The other two algorithms SIA (Spatially Inverted Algorithm) and SIA$^+$ (Advanced Spatially Inverted Algorithm) employ a Spatial Inverted Index to store the set of objects of reference. In all algorithms, the objects of interest $p \in P$ are stored in a R*-tree [Beckmann et al. 1990]. First, we present the algorithms considering the range as the spatial selection criteria; next we adapt each algorithm to consider the nearest neighbor selection criteria. Before presenting the algorithms, we briefly describe the Adapted Inverted File and the Spatial Inverted Index structures used in the algorithms.

Adapted Inverted File. An Inverted File [Zobel and Moffat 2006] maps each term of a vocabulary to an Inverted List, containing the documents (text) that have the term. For each entry in the list, the *id* of the document and the frequency (or the *term impact*[2]) of the document is stored [Anh et al. 2001; Rocha-Junior et al. 2011; Salton and Buckley 1988; Zobel and Moffat 2006]. Therefore, given a term, the list of documents that contains the term can be obtained efficiently. IFA employs an adapted Inverted File to store the spatio-textual objects of reference $f \in F$. Besides the *id* of f and the term impact, each entry stores the spatial location of f. Thus, it is more efficient to check, at query time, if a given object of reference f attends the spatial selection criteria.

Spatial Inverted Index. The Spatial Inverted Index (S2I) is a hybrid index structure to process spatio-textual queries [Rocha-Junior et al. 2011; Chen et al. 2013]. The index can search spatio-textual data in an optimized way. Similar to an Inverted File, the S2I stores for each term of a vocabulary, the set of objects that contains the term. However, different from an Inverted File, the S2I stores the most frequent terms in a Spatial Index (aR-tree [Papadias et al. 2001]), while the less frequent terms are stored in an Inverted List. Each entry of the S2I stores for each object the id, the spatial location and the term impact.

4.1 Inverted File based Algorithm

The naive approach to process the Top-k Spatial Keyword Preference Query requires finding all spatio-textual objects in the vicinity of each spatial objects of interest and computing the textual relevance of each spatio-textual object to the query keywords, in order to compute the score of each spatial object of interest. These steps have to be repeated for all object of interest, in order to find the k best objects. This approach is inefficient.

The idea behind the Inverted File based Algorithms (IFA) is to filter the objects that are not textually relevant by using the Inverted File, reducing the processing cost. Specifically, the spatio-textual objects of reference that are not present in the inverted lists of the terms t in $Q.D$ are not relevant to the query keywords $Q.D$.

The objects are retrieved from the Inverted Lists ordered by the *id*, which improves the merging process to compute the final textual relevance of an object f. Once an object f is found in one inverted list of a term t_i, the entry contains the impact of the term t_i in the textual relevance of f. Computing the textual relevance of f for all terms in $Q.D$ requires retrieving the same object of reference f from all inverted lists. The textual score of f is the sum of all partial scores of f retrieved from each inverted list. After computing the score of each object of reference f, the algorithm checks if f attends the spatial selection criteria. In the case of range, the criteria is attended if $dist(p, f) \leq Q.r$. Finally, the textual score of f is attributed to p if the score of f is higher than the current score of p ($p.score$), ($\theta(f.D, Q.d) > p.score$). The k objects of interest with the highest scores are maintained in a heap of k size. After computing the score of all objects p, the heap contains the k best objects of interest.

Algorithm 1 presents the IFA algorithm. The algorithm computes the score of each object $p \in P$ (lines 3-27), initially the score of p is zero (line 4). It then employs an iterator (line 6) to access all objects in the Inverted List of a given term t. The objects are accessed in increasing order of *id*. The algorithm employs the heap H ordered by *id* to store the references for the unvisited objects of the inverted lists. Each entry e of the heap H has two attributes $e.f$ and $e.l$ ($e = \{e.f, e.l\}$). The $e.f$ is the next object in the list to be visited, while $e.l$ is a reference to the list iterator (lines 7-9). The objective of the heap H is to store, for each inverted list of a term $t \in Q.D$, the spatio-textual object of reference f with smallest *id* that is textually relevant to the term t that has not being visited yet.

[2]The term impact is the textual relevance of a term in a document, without considering the other documents of the collection. The term impact takes into account the length of the document and can be used to compare the textual relevance of two different documents according to a single term t that they have in common [Anh et al. 2001; Rocha-Junior et al. 2011; Salton and Buckley 1988].

Algorithm 1: *Inverted File based Algorithm* (IFA)

Input: $Q = (Q.D, Q.r, Q.k)$ //The $Q.r$ (radius) is the spatial selection criteria for range (rng).
Output: Heap that maintains the k best objects of interest.

```
1  M ← ∅ //Heap that maintains the k best objects of interest.
2  H ← ∅ //Heap that maintains the entries e ordered by the id of the object of reference f.
3  for each p ∈ P do
4  |  p.score ← 0
5  |  for each t ∈ Q.D do
6  |  |  iterator ← IF.list(t).iterator()
7  |  |  if iterator.hasNext() then
8  |  |  |  H.add({iterator.next(), iterator})
9  |  |  end
10 |  end
11 |  e ← nextEntry(H)
12 |  while e ≠ null AND H ≠ ∅ do
13 |  |  e' ← nextEntry(H)
14 |  |  while e' ≠ null AND e.f = e'.f do
15 |  |  |  e.f.θ ← e.f.θ + e'.f.θ
16 |  |  |  e' ← nextEntry(H)
17 |  |  end
18 |  |  updateScore(p, e.f)
19 |  |  e ← e'
20 |  end
21 |  updateScore(p, e.f)
22 |  if |M| < k OR p.score > M.peekMin().score then
23 |  |  M.add(p)
24 |  |  if |M| > k then
25 |  |  |  M.removeMin()
26 |  |  end
27 |  end
28 end
29 return M
```

The algorithm resumes accessing the next entry in the list. This entry has the spatio-textual object of reference e.f has the smallest id (line 11). The function $nextEntry(H)$ removes the entry e from the Heap ($e \leftarrow heap.pollFirst()$) and adds a new entry $\{e.l.next(), e.l\}$ with the next object f in the same list. If the heap is empty, the entry e will be $null$. While e is not $null$ and H is not empty (lines 12-20), the algorithm computes the score of p. The score of p is the highest score among the spatio-textual objects of reference f that are textually relevant and that attend the spatial selection criteria. Thus, before setting the score of p, the textual score of f is computed. The score of f is the sum of the partial scores of f found in each Inverted List in which it appears. Since the lists are ordered by id of the objects of reference, the score of f is computed by checking if f is in the next object to be accessed in all inverted lists (lines 14-17).

For example, assuming that the query keywords $Q.D$ contains two terms t_1 and t_2, $Q.D = \{t_1, t_2\}$ and the spatio-textual object of reference f_1 has in its textual description $f.D$ the same terms t_1 and t_2. In this case, the object f_1 appears in both Inverted Lists of t_1 and t_2. Once the objects are accessed in the list in increasing order of id, f_1 (whose $id = 1$) is in the top of both lists. In order to compute the score of f_1, both entries are retrieved from the heap and the partial scores in each list is summed to obtain the final score of f_1. On the other hand, if f_1 had only the term t_1, it would not

Algorithm 2: *Spatially Indexed Algorithm* (SIA)

Input: $Q = (Q.D, Q.r, Q.k)$ //The $Q.r$ (radius) is the spatial selection criteria for range (rng).
Output: Heap that maintains the k best objects of interest.

```
1  M ← ∅ //Heap that maintains the k best objects of interest.
2  H ← ∅ //Heap that maintains the entries e ordered by the id of the object of reference f.
3  for each p ∈ P do
4  |  p.score ← 0
5  |  for each t ∈ Q.D do
6  |  |  iterator ← S2I.search(t, p.x, p.y, Q.r)
7  |  |  lines 7-9 of the IFA algorithm
8  |  end
9  |  lines 11-27 of the IFA algorithm
10 end
11 return M
```

appear in the Inverted List of t_2. Thus, the second entry in the heap would be of another object of reference with a higher id. In this case, the final score of f_1 would be the partial score found in the Inverted List of t_1.

After computing the score of the spatio-textual object of reference f, the function *updateScore(p, e.f)* updates the score of p. The function checks if f attends the spatial selection criteria and if the textual score of f is higher than the current score of p, updating the score of p if both conditions are true. After computing the score of p, the algorithm updates the heap M that maintains the k objects with the highest scores (lines 22-27). Therefore p is added into M only if M has less than k objects or if the score of p is higher than the smallest score among the objects currently stored in M ($p.score > M.peekMin().score$). The object in M with the smallest score is removed when the size of M is larger than k (lines 24-26). The algorithm repeats this process until the heap H is empty or $e = null$, returning the k objects with the highest scores stored in M.

4.1.1 ***Nearest Neighbor***. Few changes are required to adapt IFA (Algorithm 1) for processing the Top-k Spatial Keyword Preference Queries by assuming the nearest neighbor as the spatial selection criteria ($Q.\psi = nn$). First, the algorithm receives a new variable named $minDist$ (line 3) to maintain the minimum distance between p and a relevant spatio-textual object of reference $e.f$ ($e.f.\theta > 0$). The variable $minDist$ is initialized with the largest distance among two objects in the dataset. The $minDist$ is updated every time that a new relevant object of reference $e.f$ is found with a smaller distance to p than $minDist$. This update happens in the *updateScore(p, e.f)* function. Second, the score of p is updated when one of the following conditions happens: 1) a textually relevant spatio-textual object of reference $e.f$ with a smaller distance to p is found; or 2) the distance between $e.f$ and p has the same smallest distance, but the textual score of $e.f$ is higher than the current score of p. This update also happens in the *updateScore(p, e.f)* function.

4.2 Spatially Indexed Algorithm

The *Spatially Indexed Algorithm (SIA)* employs a hybrid index, instead of an Inverted File, for processing the query. Similar to IFA, SIA also computes the score of each $p \in P$, before finding the top-k objects. However, different from IFA that filters the objects that are textually irrelevant only, SIA filters the objects that are textually and spatially irrelevant.

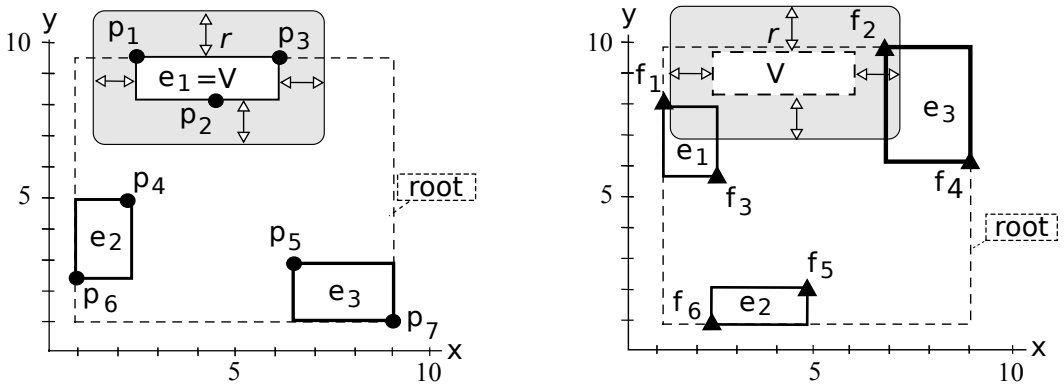
Algorithm 2 presents SIA. The algorithm computes the score of each object $p \in P$, initially the score of p is zero (line 4). For each term $t \in Q.D$, the algorithm accesses the $S2I$ in order to get an iterator that accesses the spatio-textual objects of reference f in increasing order of id. Only the

Algorithm 3: *Optimized Spatially Indexed Algorithm* (SIA+)

```
   Input: Q = (Q.D, Q.r, Q.k) //The Q.r (radius) is the spatial selection criteria for range (rng).
   Output: Heap that maintains the k best objects of interest.
1  M ← ∅ //Heap that maintains the k best objects of interest.
2  H ← ∅ //Heap that maintains the entries e ordered by the id of the object of reference f.
3  for each V ∈ P do
4  |  ∀p ∈ V, p.score ← 0
5  |  for each t ∈ Q.D do
6  |  |  iterator ← S2I.search(t, V.MBR, Q.r)
7  |  |  lines 7-9 of the IFA algorithm
8  |  end
9  |  e ← nextEntry(H)
10 |  while e ≠ null AND H ≠ ∅ do
11 |  |  e' ← nextEntry(H)
12 |  |  while e' ≠ null AND e.f = e'.f do
13 |  |  |  e.f.θ ← e.f.θ + e'.f.θ
14 |  |  |  e' ← nextEntry(H)
15 |  |  end
16 |  |  updateScore(V, e.f) e ← e'
17 |  end
18 |  updateScore(V, e.f)
19 |  for each p ∈ V do
20 |  |  if |M| < k OR p.score > M.peakMin().score then
21 |  |  |  M.add(p)
22 |  |  |  if |M| > k then
23 |  |  |  |  M.removeMin()
24 |  |  |  end
25 |  |  end
26 |  end
27 end
28 return M
```

objects that are textually and spatially relevant are returned by the $S2I$ (line 6). The rest of the algorithm is identical to IFA.

4.2.1 ***Nearest Neighbor***. Some changes are required to adapt SIA (Algorithm 2) for processing Top-k Spatial Keyword Preference Queries by assuming the nearest neighbor as the spatial selection criteria ($Q.\psi = nn$). First, it adds the variable $minDist$ (line 3) to maintain the minimum distance between p and a relevant spatio-textual object of reference $e.f$ ($e.f.\theta > 0$). The variable $minDist$ is initialized with the largest distance between any two objects in the dataset. The variable is updated every time that a new relevant object of reference $e.f$ is found whose distance to p is smaller than the current $minDist$. The update happens in the *updateScore(p, e.f)* function. Second, instead of calling the function $S2I.search(t, p.x, p.y, Q.r)$ (line 6) that receives as parameter the radius r, the algorithm calls the function $S2I.searchNN(t, p.x, p.y)$ that returns an iterator to access all spatio-textual objects with the same smallest distance to p. Third, the algorithm updates the score of p in the *updateScore(p, e.f)* function. The score of p is updated when a relevant spatio-textual object of reference $e.f$ with a smaller distance to p is found, or when the distance between $e.f$ and p has the same smallest distance, but the textual score of $e.f$ is higher than the current score of p.



(a) Spatial objects of interest p.

(b) Spatio-textual objects of reference f.

Fig. 3. Objects of reference necessary to compute the score of objects $p \in V$.

4.3 Optimized Spatially Indexed Algorithm

The Optimized Spatially Indexed Algorithm SIA$^+$ is an extension of SIA that concurrently computes the score of a set V of spatial objects of interest that are spatially near to each other. The idea is accessing the S2I only once to compute the score of all objects $p \in V$, reducing I/O.

For example, Fig. 3 presents the spatial objects of interest p (Fig. 3(a)) and the spatio-textual objects of reference f (Fig. 3(b)). In this example, the set V is composed by the objects p_1, p_2 and p_3 and the query radius is r. Therefore, the SIA+ searches the index that stores the objects of reference for the objects that can contribute with the score computation of any object p in V, using the MBR (Minimum Bounding Box) that encapsulates all objects in V expanded with the query radius (gray area) to guarantee that all important objects are selected. Looking for the projection of the area of interest in Fig. 3(b), the only important object is f_2, because all the other objects are outside the range area of the MBR of V. Thus, f_2 is returned and its distance to all objects in V is computed to check which objects in V are in the radius distance to f_2, which is only p_3. The score of p_3 is computed based on the textual relevance of f_2 and p_3 is added in the heap if its score is higher than the score of the kth object already stored in the heap. The problem of this approach is if the MBR of V is large. For example, if V was composed by objects p_1 and p_7, it would have a large MBR. In this case, almost all features would be returned to compute the score of the objects p_1 and p_7.

Algorithm 3 presents SIA$^+$. For each $V \in P$, where each V is a set of spatial objects of interest that are spatially near each other, the algorithm sets the scores of the objects $p \in V$ to zero (line 4). Then, the algorithm accesses the S2I to select the candidate spatio-textual objects of reference that can contribute in the score computation of the object $p \in V$ (line 6). In order to find the candidate objects, the algorithm computes the distance between f and the MBR (Minimum Bounding Rectangle) that encloses the objects $p \in V$. If the distance between f and $V.MBR$ is smaller than or equal to $Q.r$, the object f is selected as candidate. Therefore, the smaller the size of the MBR, the lower the the number of candidates, reducing the cost of the algorithms in terms of query processing.

Lines 9-18 are identical to IFA and SIA, except lines 16 and 18 that instead of computing the score of a single object p, SIA$^+$ requires computing the score of all objects $p \in v$. The same happens in the lines 19-26, where the algorithm checks for each $p \in V$, if p can be inserted in the heap M that maintains the k best objects.

4.3.1 ***Nearest Neighbor***. Some changes are required to adapt SIA$^+$ (Algorithm 3) for processing the Top-k Spatial Keyword Preference Query by assuming the nearest neighbor as the spatial selection criteria ($Q.\psi = nn$). First, each $p \in V$ receives a variable $p.minDist$ (line 5) to maintain the minimum

distance between p and a relevant spatio-textual object of reference $e.f$ ($e.f.\theta > 0$). The variable $p.minDist$ is initialized with the largest distance among two objects in the dataset. The variable is updated in the *updateScore(p, e.f)* function every time that a new relevant object of reference $e.f$ is found whose distance to p is smaller than the current $p.minDist$.

Second, instead of calling the function *S2I.search(t, V.MBR, Q.r)* (line 6), the algorithm calls the function *S2I.searchNN(t, V)* that returns an iterator to access all spatio-textual candidate objects to compute the score of all objects $p \in V$. The function *S2I.searchNN(t, V)* maps each $p \in V$ to the variable $minDist$ that stores the minimum distance between p and any other feature visited during this index access. The function also maps each p to the set of objects of reference C with the same smallest distance[3]. The function also maintains a variable named $worstMinDist$ that stores the maximum minimum distance among the objects $p \in V$. This variable works as lower bound, stopping the search for candidate objects when there is no other object in the index whose distance to $V.MBR$ is smaller or equals the $worstMinDist$. After stopping the search, the function returns the spatio-textual objects of references C associated with each $p \in V$ in increasing order of id (line 6).

Third, the algorithm updates the score of p in the *updateScore(p, e.f)* function. The score of p is updated when a relevant spatio-textual object of reference $e.f$ with a smaller distance to p is found, or when the distance between $e.f$ and p has the same smallest distance, but the textual score of $e.f$ is higher than the current score of p.

In this article, we have chosen to group the objects spatially by putting in a set V the objects p that are near each other. Since the spatial objects of interest p are stored in a Spatial Index (R*-tree), we have chosen to make each leaf of the R*-Tree one set V. Another solution could be employing a cluster algorithm to create groups of objects spatially close each other, where each group would compose a set V.

4.4 Complexity Analysis of the Algorithms

In this section, we make a brief complexity analysis of the algorithms proposed for processing the Top-K Spatial Keyword Preference Query.

In the baseline algorithm, the score of each object $p \in P$ is computed comparing the distance between p and each $f \in F$ and the textual relevance of f for the query keywords. Therefore, the complexity of the baseline algorithm is $\mathcal{O}(|P| \cdot |F|)$. In the IFA algorithm, the score of each object $p \in P$ is computed comparing the distance between p and each $f' \in F'$ and the textual relevance of f', where F' is a subset of F ($F' \subseteq F$) that contains the objects of reference f' that are textually relevant to the query keywords (the objects in the inverted lists of the query keywords). Therefore, the complexity of the IFA algorithm is $\mathcal{O}(|P| \cdot |F'|)$. In the SIA algorithm, the score of each object $p \in P$ is computed checking only the textual relevance of the objects $f'' \in F''$ that are textually and spatially relevant to compute the score of p, where F'' is a subset of F' ($F'' \subseteq F' \subseteq F$) that contains the objects of reference f'' that are spatially and textually relevant to the query keywords and spatially relevant to compute the score of p. Therefore, the complexity of the SIA algorithm is $\mathcal{O}(|P| \cdot |F''|)$. Different from the other algorithm, SIA+ does not compute the score for each $p \in P$, but for a set objects $V \in P$. Similar to SIA, the SIA+ algorithm employs the set F'' of objects that are textually and spatially relevant to the set of objects V. Therefore, the complexity of SIA+ algorithm is $\mathcal{O}(\frac{|P|}{|V|} \cdot |F''|)$.

[3]This set is very small in practice, since the probability of having objects with the same smallest distance is small.

Table I. Settings of the experiments. The default values are presented in bold.

Parameters	Values
Number of results (k)	1, **5**, 10, 15
Number of keywords	1, **3**, 5, 7
Datasets	Venice, **London**, North America

5. EXPERIMENTAL EVALUATION

In this section, we evaluate the algorithms proposed (IFA, SIA e SIA$^+$). All algorithms were implemented in Java, using the XXL Library[4]. All experiments were executed in the same machine with an Intel Processor of 2.0GHz and 4GB of RAM memory.

In each experiment, we evaluate the impact of a single variable, while the others are maintained fixed. The variables studied are I/O (pages of 4MB accessed from the disk), response time and group size (in the case of SIA$^+$). The response time is measured in milliseconds. We repeat the same experiment 10 times and collect the average results. In each round, we execute the query 50 times. The terms used in each experiment are from a set with the 500 most frequent terms. The radius, for the range selection criteria, is set in approximately[5] 200m.

Table I presents the settings of the experiments. The values in bold are the default values, when not explicitly mentioned. All figures are in logarithm scale due to the huge difference in the performance of the algorithms.

The remaining of this section is organized as follows: first, we present the datasets used in the experiments; next, we study the performance of the algorithms in terms of I/O and response time, while varying the number of query keywords, the number or results and datasets. Finally, we evaluate the impact of varying the size of the groups V in the performance of SIA+ algorithm. We present the results for the range spatial selection criteria ($Q.\psi = rng$), since the results for nearest neighbor selection criteria were similar.

5.1 Datasets

The datasets used in the experiments were obtained at Mapzen[6] and GEOFABRIK[7]. These sites maintain extracts from the OpenStreetMap for the main cities and countries, obtained from OpenStreetMap (http://www.osm.org). The datasets with data from Venice and London were obtained at Mapzen, while the dataset of the North America was obtained at GEOFABRIK.

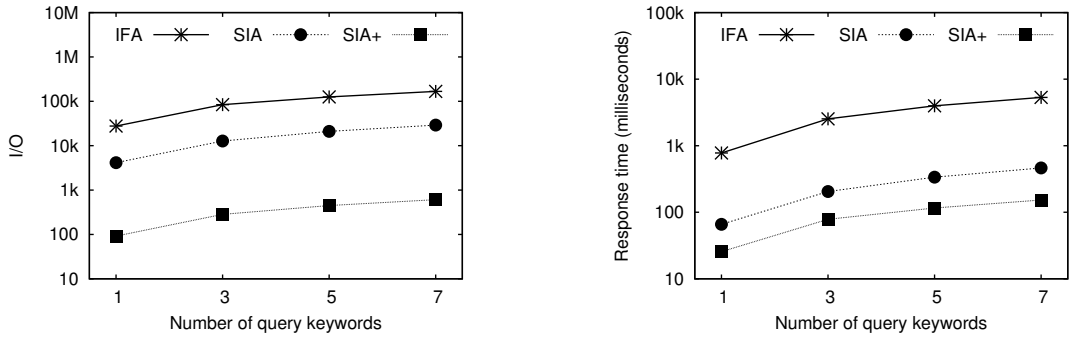
We process the datasets to extract only the spatio-textual objects. The set of objects of interest P is composed by spatial objects whose the category in the OpenStreetMap is hotel, while the set of spatio-textual objects of reference F is composed by the other spatio-textual objects.

Table II presents some characteristics of the datasets: the number of objects of interest $|P|$, the number of spatio-textual objects of reference $|F|$, the number of unique terms in the dataset and the total number of terms. The datasets used in the experiments can be downloaded from https://goo.gl/zHExTn.

[4]http://dbs.mathematik.uni-marburg.de/Home/Research/Projects/XXL
[5]The function employed to compute the distance does not consider the inclination of the planet earth. The same function is employed by all algorithms, without interfering in the results obtained.
[6]http://mapzen.com/metro-extracts
[7]http://download.geofabrik.de/

Table II. Some characteristics of the datasets used in the experimental evaluation.

Datasets	$\|P\|$	$\|F\|$	Number of unique terms	Total number of terms
Venice	504	167,958	14,678	408,747
London	1,341	463,066	56,569	1,198,649
North America	9,132	2,521,344	187,179	8,881,870



(a) I/O. (b) Response time.

Fig. 4. Impact on I/O and response time, while varying the number of query keywords (k).

5.2 Varying the number of query keywords

In this experiment, we study the impact in I/O and response time in the algorithms proposed, while varying the number of query keywords (Fig. 4).

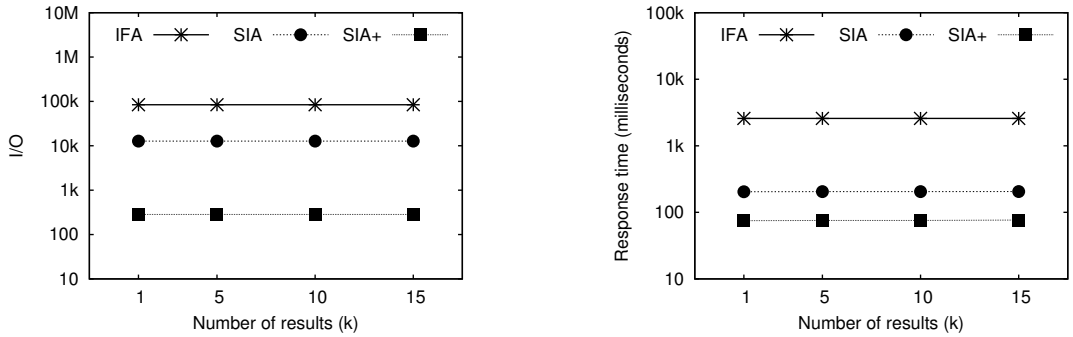
SIA$^+$ is better than the other algorithms in terms of I/O and response time for all setups. The performance of SIA$^+$ in terms of I/O (Fig. 4(a)) is more than one order of magnitude better than the performance of SIA, SIA+ is 4,497% better than SIA in terms of I/O for three query keywords. This shows the efficacy of accessing the index in groups. SIA$^+$ also presented better result in terms of response time (Fig. 4(b)), SIA+ is 261% better than SIA in terms of response time for three query keywords. The results in terms of I/O are not completely traduced in the response time. The main reason is the number of false positive spatio-textual objects of reference that are selected searching for V in the Spatial Inverted Index.

Varying the number of keywords has a significant impact in the I/O and response time. The more the number of keywords, the more the number of data that must be accessed in query time. Thus, all algorithms were impacted by the change in this variable.

5.3 Varying the number of results

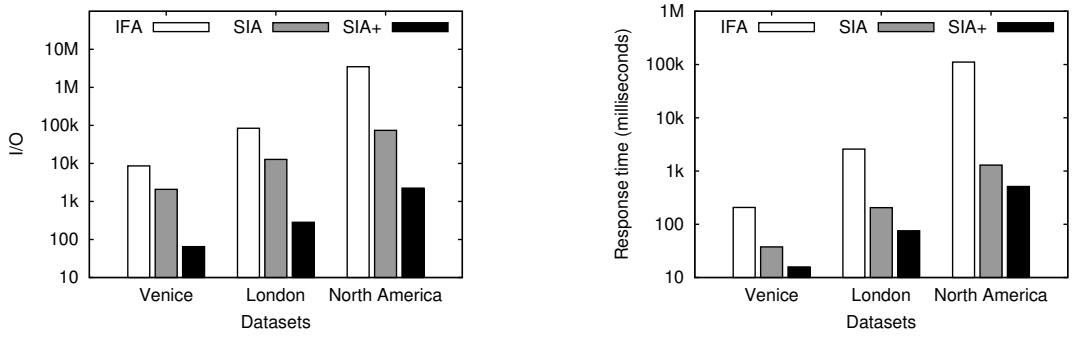
In this section, we study the impact on I/O and response time, while varying the number of results in the performance of the algorithms proposed (Fig. 5).

SIA$^+$ is better than the other algorithms in terms of I/O and response time for all setups. The performance of SIA$^+$ in terms of I/O (Fig. 5(a)) is more than one order of magnitude better than the performance of SIA, SIA+ is 4,497% better than SIA in terms of I/O for all values of k. This shows the efficacy of the group access strategy in reducing the number of times the index is accessed, and consequently the I/O. The response time of SIA$^+$ is also better (Fig. 5(b)), SIA+ is 271% better than SIA in terms of response time for $k = 5$. However, the difference between SIA and SIA$^+$ in terms of response time is smaller than the difference in terms of I/O. The group processing reduces I/O, but the number of false positive spatio-textual objects of reference that are selected due to the search for



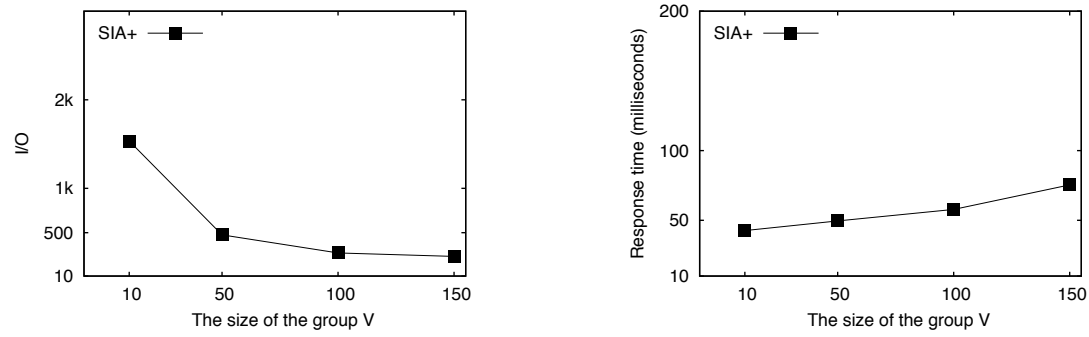
(a) I/O. (b) Response time.

Fig. 5. Impact on I/O and response time, while varying the number of results (k).



(a) I/O. (b) Response time.

Fig. 6. Impact on I/O and response time for different datasets.



(a) I/O. (b) Response time.

Fig. 7. Impact on I/O and response time, while varying the size of the groups V for SIA$^+$ algorithm.

V causes an additional cost in terms of processing. Therefore, the importance of selecting groups of objects that are very close each other.

Varying the number of results has no impact in the I/O (Fig. 5(a)) and almost no impact in the response time 5(b)). All algorithms proposed (IFA, SIA and SIA$^+$) calculate the score of each $p \in P$, maintaining the k best objects in the heap. Therefore, for the same keywords, the same algorithm will access the same number of disk pages independent of the size of k, incurring in no impact in terms of I/O. However, varying the number of k may affect the response time, since the size of the

heap increases and more processing steps are required to maintains this heap updated with the k best results. However, the heap maintains few items (from 1 to 15) and they are kept in main memory during the query processing. Therefore, increasing the value of k has almost no additional cost in terms of response time.

5.4 Varying the datasets

In this experiment, we study the impact in the I/O and response time for different datasets (Fig. 6).

Fig. 6(a) presents the results in terms of I/O. SIA$^+$ is better than the other algorithms in terms of I/O for all datasets, SIA+ is 3,201%, 4,497% and 3,310% better than SIA in Venice, London and North America datasets respectively. In this experiment, the difference between SIA and IFA is smaller when compared with the other experiments. The main reason is that SIA employs an hybrid index that stores the inverted lists of the frequent term in a spatial index. Therefore, the bigger the dataset the larger the number of the Inverted Lists that are stored in the spatial indexing, giving a bigger advantage for SIA and SIA$^+$ in terms of I/O compared to IFA.

Fig. 6(b) presents the impact in the response time. SIA+ is 237%, 272% and 252% better than SIA in Venice, London and North America datasets respectively. The difference is consistent among the datasets, indicating that more populated datasets do not have a significant impact in increasing or reducing the number of false positives, showing the efficacy of SIA+ for small and large datasets.

5.4.1 *Varying the size of the groups.* In this experiment, we study the impact of the size of the groups V in the I/O and response time of SIA$^+$ (Fig. 7).

The larger the group, the smaller the number of times the index is accessed, since one access is sufficient to select the spatio-textual candidates to process the score of all objects $p \in V$. Therefore, as shown in Fig. 7(a), the I/O reduces when the size of the group increases. On the other hand, the larger the group, the bigger the MBR that encapsulates all objects in the group, selecting more false positives candidates. As presented in Fig. 7(b), the smaller the size of V, the better the response time.

The size of the group V can be employed to adjust SIA$^+$ to attend different objectives of an application. If the main objective is to access less data, the number of elements in V should be increased. However, if the objective is to reduce the response time, smaller V is better.

The success of SIA$^+$ depends on the groups created. If the objects in a group are very close each other (which has a higher probability of happening when the group is small), the number of candidates is reduced, and the query process has good results. However, if there are some objects distant of each other, the MBR of V will be large and SIA$^+$ will select a lot of false candidates, leading to poor performance.

The default value of V employed in the other experiments is 102. Therefore, SIA$^+$ could have presented better performance in terms of response time if we had selected the size of V to be 50, for instance.

6. FINAL REMARKS

In this article, we have presented a new query type named Top-k Spatial Keyword Preference Query. This query selects the k objects of interest considering other spatio-textual objects in their vicinity that are relevant to the query keywords. We have also presented three algorithms to process this query IFA, SIA and SIA$^+$. The IFA algorithm employs an Inverted File to reduce the number of objected accessed at query time, while SIA and SIA$^+$ have employed a spatio-textual index. SIA and SIA$^+$ have performed better than IFA in all setups, showing the importance of employing hybrid indexes

to process this query. SIA$^+$ had better performance than SIA, mainly in terms of I/O due to a group processing technique that accesses the index only once to obtain the score of a set of objects of interest concurrently. We have also observed that the group size has significant impact in terms of I/O and response time.

In the future, we plan to develop a new algorithms to process the Top-k Spatial Keyword Preference Query for the influence selection criteria. The SIA and SIA$^+$ cannot be adapted to process queries with this spatial selection criteria, because the influence has no spatial limit such as the range and the nn. Other directions are processing this query by employing the network distance, instead of the Euclidean distance. In this case, other indexes and algorithms are required. Another interesting direction is to study how to process this query in a distributed scenario. Finally, it is very important to evaluate this query qualitatively in order to evaluate its relevance compared with the Traditional Top-k Spatial Preference Query.

REFERENCES

Anh, V. N., de Kretser, O., and Moffat, A. Vector-space Ranking with Effective Early Termination. In *Proceedings of the International ACM SIGIR Conference on Research & Development of Information Retrieval*. New Orleans, USA, pp. 35–42, 2001.

Beckmann, N., Kriegel, H.-P., Schneider, R., and Seeger, B. The R*-tree: an efficient and robust access method for points and rectangles. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. New York, USA, pp. 322–331, 1990.

Cao, X., Chen, L., Cong, G., Jensen, C. S., Qu, Q., Skovsgaard, A., Wu, D., and Yiu, M. L. Spatial Keyword Querying. In P. Atzeni, D. Cheung, and S. Ram (Eds.), *Conceptual Modeling*. Lecture Notes in Computer Science, vol. 7532. Springer, pp. 16–29, 2012.

Chen, L., Cong, G., Jensen, C. S., and Wu, D. Spatial Keyword Query Processing: an experimental evaluation. *Proceedings of the VLDB Endowment* 6 (3): 217–228, 2013.

Cong, G., Jensen, C. S., and Wu, D. Efficient Retrieval of the Top-k Most Relevant Spatial Web Objects. *Proceedings of the VLDB Endowment* 2 (1): 337–348, 2009.

De Felipe, I., Hristidis, V., and Rishe, N. Keyword Search on Spatial Databases. In *Proceedings of the IEEE International Conference on Data Engineering*. Cancun, Mexico, pp. 656–665, 2008.

Papadias, D., Kalnis, P., Zhang, J., and Tao, Y. Efficient OLAP Operations in Spatial Data Warehouses. In C. S. Jensen, M. Schneider, B. Seeger, and V. J. Tsotras (Eds.), *Advances in Spatial and Temporal Databases*. Lecture Notes in Computer Science, vol. 2121. Springer, pp. 443–459, 2001.

Rocha-Junior, J. B., Gkorgkas, O., Jonassen, S., and Nørvåg, K. Efficient Processing of Top-k Spatial Keyword Queries. In D. Pfoser, Y. Tao, K. Mouratidis, M. A. Nascimento, M. Mokbel, S. Shekhar, and Y. Huang (Eds.), *Advances in Spatial and Temporal Databases*. Lecture Notes in Computer Science, vol. 6849. Springer, pp. 205–222, 2011.

Rocha-Junior, J. B., Vlachou, A., Doulkeridis, C., and Nørvåg, K. Efficient Processing of Top-k Spatial Preference Queries. *Proceedings of the VLDB Endowment* 4 (2): 93–104, 2010.

Salton, G. and Buckley, C. Term-weighting Approaches in Automatic Text Retrieval. *Information Processing and Management* 24 (5): 513–523, 1988.

Tsatsanifos, G. and Vlachou, A. On Processing Top-k Spatio-textual Preference Queries. In *Proceedings of the International Conference on Extending Database Technology*. Brussels, Belgium, pp. 433–444, 2015.

Vaid, S., Jones, C. B., Joho, H., and Sanderson, M. Spatio-textual Indexing for Geographical Search on the Web. In C. B. Medeiros, M. J. Egenhofer, and E. Bertino (Eds.), *Advances in Spatial and Temporal Databases*. Lecture Notes in Computer Science, vol. 3633. Springer, pp. 218–235, 2005.

Yiu, M. L., Dai, X., Mamoulis, N., and Vaitis, M. Top-k Spatial Preference Queries. In *Proceedings of the IEEE International Conference on Data Engineering*. Istanbul, Turkey, pp. 1076–1085, 2007.

Zhou, Y., Xie, X., Wang, C., Gong, Y., and Ma, W.-Y. Hybrid Index Structures for Location-based Web Search. In *Proceedings of the International Conference on Information and Knowledge Engineering*. Bremen, Germany, pp. 155–162, 2005.

Zobel, J. and Moffat, A. Inverted Files for Text Search Engines. *ACM Computing Surveys* 38 (2): 6:1–6:56, 2006.