The VLDB Journal (2018) 27:445–470
https://doi.org/10.1007/s00778-018-0504-y

REGULAR PAPER

Finding the optimal location and keywords in obstructed and unobstructed space

Farhana Murtaza Choudhury[1] · J. Shane Culpepper[1] · Zhifeng Bao[1] · Timos Sellis[2]

Received: 27 May 2017 / Revised: 12 January 2018 / Accepted: 4 April 2018 / Published online: 25 April 2018

Abstract
The problem of optimal location selection based on reverse k nearest neighbor (RkNN) queries has been extensively studied in spatial databases. In this work, we present a related query, denoted as a *Maximized Bichromatic Reverse Spatial Textual k Nearest Neighbor* (MaxST) query, that finds an optimal location and a set of keywords for an object so that the object is a kNN object for as many users as possible. Such a query has many practical applications including advertisements, where the query is to find the location and the text contents to include in an advertisement so that it is relevant to the maximum number of users. The visibility of the advertisements also has an important role in the users' interests. In this work, we address two instances of the spatial relevance when ranking items: (1) the Euclidean distance and (2) the visibility. We carefully design a series of index structures and approaches to answer the MaxST for both instances. Specifically, we present (1) the GRP- TOPK approach that requires the computation of the top-k objects for all of the users first and then applies various pruning techniques to find the optimal location and keywords; (2) the INDIV- U approach, where we use similarity estimations to avoid computing the top-k objects of the users that cannot be a final result; and (3) the INDEX- U approach where we propose a hierarchical index structure over the users to improve pruning. We show that the keyword selection component in MaxST queries is NP-hard and present both approximate and exact solutions for the problem.

Keywords Spatial–textual query · Reverse kNN · Efficiency

1 Introduction

The optimal location selection problem is an important task in making business decisions. As a result, a number of studies have addressed different instances of this problem, and queries such as the *Maximized Bichromatic Reverse k Nearest Neighbor* (MaxBRkNN) queries [20,33,34,43] have received considerable attention in the spatial database community in recent years. Given a set of users U and a set of objects O over a shared spatial dataspace, a bichromatic reverse k nearest neighbor (BRkNN) query for an object $o \in O$ returns all the users $u \in U$ for which o is a kNN in O. A MaxBRkNN query finds the optimal region to place a new object, $p \notin O$ such that the number of users in the result of the BRkNN query issued by p is maximized.

A practical application of these queries is to find the location of a new business or a new facility that can serve the maximum number of customers. In the literature, spatial distance is usually the only relevance criteria considered. However, customers are generally interested in the products and services as well as the location. Therefore, spatial–textual query is an interesting extension in this setting. The problem has recently been studied for spatial–textual databases, where [14] find a set of keywords for an object in a fixed location such that the object is a kNN of the maximum number of users w.r.t. both spatial and textual similarities.

Despite significant progress on this problem, there is a research gap in finding both an optimal location and a keyword set for an object, which is a natural extension of the problem. Moreover, as targeted applications become more practical, physical obstacles must also be factored into the

✉ Farhana Murtaza Choudhury
farhana.choudhury@rmit.edu.au

J. Shane Culpepper
shane.culpepper@rmit.edu.au

Zhifeng Bao
zhifeng.bao@rmit.edu.au

Timos Sellis
tsellis@swin.edu.au

[1] RMIT University, Melbourne, Australia

[2] Swinburne University of Technology, Hawthorn, Australia

solution. For example, visibility can have an important role in improving advertisement reach, frequency, and impact. While previous research has explored various visibility-related queries, no prior work has considered the effect of visual obstacles in the context of BRkNN problems. Consider the following example applications that highlight the importance of these factors.

Example 1 In social media advertising, a user is shown a limited number of advertisements that are highly relevant to their location and preferences (top-k relevant advertisements). In this case, given a set of candidate locations and keyword choices, the application is to find the location and a limited number of keywords to include in an advertisement such that it is displayed to the maximum number of users.

Example 2 Consider a company who wants to place a new billboard for an advertisement p. An important attribute in this scenario is the visibility of the billboard for potential customers in the presence of obstacles, such as buildings in a city. So given a set of potential locations where a billboard can be placed and a set of keywords appropriate for the advertisement, the problem is to find an optimal location and a limited number of keywords for p such that p is relevant and visually unobstructed for as many customers as possible.

In both examples, the underlying problem is to find the location and text for a specific product or service such that the product is one of the k most relevant objects of the maximum number of users, which is maximizing the size of the bichromatic reverse kNN of the product. Here, the ranking is based on both spatial and textual properties. In the first example, the spatial relevance of an object (advertisement) is the spatial proximity (Euclidean distance) from a user. In the second instance, the spatial relevance is the visibility of an object from a user in the presence of visual obstacles (buildings). In contrast to the spatial proximity, the visibility of an object from a user depends on other objects in the dataset that are located in between the user and the object.

In this article, a new query variation is presented—the *Maximized Bichromatic Reverse Spatial Textual k Nearest Neighbor* (MaxST) query. Two different instances of spatial relevance are explored for ranking objects: (i) the Euclidean distance and (ii) the visibility. A series of carefully designed indexes and traversal algorithms are proposed to process MaxST queries for both of the instances.

This article builds on our previous work [8], where the problem for the first instance (Euclidean distance as spatial relevance) was initially introduced. In that prior work, the GRP- TOPK approach was proposed to efficiently find the best location and keyword set combination that requires the computation of the top-k objects for all of the users. A new indexing structure, the MIR-tree, was also introduced. The *Modified IUR-tree* (MIUR-tree) was used to index the users and was motivated by the desire to avoid computing the top-k objects for users that cannot be in the final result set.

The visibility of an object introduces additional challenges when computing spatial relevance in this problem, as the visibility of an object o w.r.t. a user u depends not only on their own locations, but also on the other objects (visual obstacles) between o and u. Due to the importance of visibility for many applications, we extend our MaxST query solutions for visibility, which represents the first work addressing the effect of visual obstacles on Maximized Bichromatic Reverse Spatial Textual kNN queries.

In particular, this work extends our prior contributions in the following aspects—(i) the solutions from prior work [8] are extended to address the MaxST problem for both unobstructed space and obstructed space, where the spatial relevance of an object is based on Euclidean distance and visibility, respectively. (ii) In addition to the two approaches (the GRP- TOPK approach and the INDEX- U approach) presented by [8], a new approach is proposed, INDIV- U which also avoids computing the top-k objects for the users that cannot affect the final result, and does not require the set of users to be indexed. (iii) A new index structure, the OIR-tree, is proposed to support the visibility computation and extend the approaches for visibility as the spatial relevance in this problem; and (iv) a comprehensive experimental study is presented to demonstrate the efficiency of our proposed techniques and compare relative performance.

For the rest of the article, the phrase "reverse k nearest neighbor (RkNN)" is used instead of "bichromatic spatial–textual reverse k nearest neighbor" when the context is clear. We also use the terms "top-k" and "kNN" interchangeably.

2 Problem formulation

Let D be a bichromatic dataset, where U is a set of users and O is a set of objects. Each object $o \in O$ is a pair $(o.l, o.d)$, where $o.l$ is a geo-spatial position (point, rectangle, or polygon) and $o.d$ is a set of keywords (which can be empty). Each user $u \in U$ is also defined as a similar pair $(u.l, u.d)$. For an object o, let $\mathcal{B}_o$ denote the set of users that have o as its kNN based on a combined spatial and textual relevance score. The necessary notations are listed in Table 1.

Definition 1 A MaxST query $q(p, L, W, \omega, k)$ over D, where $p \notin O$ is a specific spatial–textual object, L is a set of spatial candidate locations (point, line, rectangle, etc.), W is a set of candidate keywords, ω is a positive integer where $\omega \leq |W|$, and k is the number of relevant objects to be considered, finds a $\ell \in L$ and a set of keywords $W' \subseteq W$, $|W'| \leq \omega$ such that if $p.l = \ell$ and $p.d = W'$, the cardinality $|\mathcal{B}_p|$ is maximized. If p has any existing text description, then $p.d = (W' \cup p.\mathrm{d})$ and $p.l = \ell$ combinedly maximize $|\mathcal{B}_p|$.

Table 1 Basic notation

Symbol	Description
O (U)	The set of objects (users)
L (W)	The set of candidate spatial positions (keywords)
ω	The maximum number of keywords to select
u^+	The super-user
$\mathrm{CS}(o,u)$	The combined similarity measure of an object o w.r.t. a user u
$\mathrm{SS}(o,u)$ ($\overline{\mathrm{SS}}(o,u)$)	The spatial similarity between o and u as the Euclidean distance (as visibility)
$\mathrm{TS}(o,u)$	The textual similarity between o and u
$\mathrm{VL}(o,u)$	The visible length of o from u
$\mathcal{R}_k(u)$	The similarity value of the kth ranked object of u
$\mathcal{B}_p$	The set of users that are RkNN of an object p
$\mathcal{B}^{\downarrow}{}_{\ell}$ ($\mathcal{B}^{\uparrow}{}_{\ell}$)	The set of users that are definitely (that can be) a reverse kNN of an object with location ℓ based on a lower (an upper) bound
$\mathrm{CS}^{\uparrow}(E,u)$ ($\mathrm{CS}^{\downarrow}(E,u)$)	The upper (lower) bound of similarity measure of a node E w.r.t. a user u
$\mathrm{CS}^{\uparrow}(\ell,u)$ ($\mathrm{CS}^{\downarrow}(\ell,u)$)	The upper (lower) bound of similarity measure of an object o with location $o.l=\ell$ w.r.t. a user u
LO_u	A min-priority queue to keep the k objects with the best lower bounds w.r.t. u found so far
EU	A node of an MIUR-tree of the users
Section 8	
ζ	A cell of the auxiliary Quadtree
$OR(u)$	The obstructed region of u
$OL(\zeta)$	The users for which ζ is completely inside $OR(u)$
$len^{\downarrow}(E)$, $len^{\uparrow}(E)$	The minimum (maximum) length of an object stored in the subtree rooted at node E
$o.\theta$	The angle of o w.r.t. the Cartesian X-axis
$\chi_{E,u}$, $\hat{\chi}_{E,u}$	The set of leaf level cells of the Quadtree that are visible (not visible) from u and intersects with at least one object in node E
$\mathrm{VL}^{\downarrow}_{\Delta}(\Delta o.l,u.l)$, $\mathrm{VL}^{\uparrow}_{\Delta}(\Delta o.l,u.l)$	Minimum (maximum) perceived length of segment $\Delta o.l$ from $u.l$
$\angle^{\downarrow}(\Delta o.l,u.l)$, $\angle^{\uparrow}(\Delta o.l,u.l)$	Minimum (maximum) angle between the segment Δloc_o and $u.l$
$\angle^{\downarrow}(\Delta o.l,u^+)$, $\angle^{\uparrow}(\Delta o.l,u^+)$	Minimum (maximum) angle between the segment Δloc_o and any user u from u^+

In this work, the problem of MaxST is addressed for the two instances of the spatial relevance in the ranking of an object: (i) the spatial proximity (Euclidean distance) w.r.t. a user and (ii) the visibility from a user in the presence of visual obstacles. Now we present the similarity measures that are used in this article.

Combined similarity measure: An object o is ranked based on a combined score of spatial and textual relevance with respect to a user u. Without loss of generality, the following widely adopted linear weighted combination score [9,21] is used in this article:

$$\mathrm{CS}(o,u)=\alpha\cdot\mathrm{SS}(o.l,u.l)+(1-\alpha)\cdot\mathrm{TS}(o.d,u.d)\,, \quad (1)$$

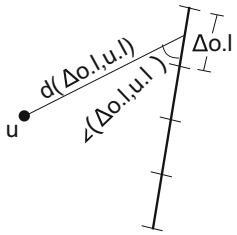
where $\mathrm{SS}(o.l,u.l)$ is the spatial similarity between the locations, the textual relevance is $\mathrm{TS}(o.d,u.d)$, and the preference parameter $\alpha\in[0,1]$ defines the importance of one relevance measure relative to the other. The value of both measures is normalized within $[0,1]$. Here, a higher score denotes higher relevance. The text similarity and two different spatial similarity measures (proximity and visibility) used in this article are now presented.

a. Text similarity: An object o is considered similar to a user u iff $o.d$ contains at least one term $t\in u.d$. Several measures can be used to compute the similarity between any two text descriptions [23]. We use the TF·IDF metric [30] for illustration purpose in this work, but our approach is applicable to any text-based similarity measure.

The term frequency, $TF(t,o.d)$, counts how many times the term t appears in a document object $o.d$, and the Inverse Document Frequency, $IDF(t,O)=\log\frac{|O|}{|o\in O,TF(t,o.d)>0|}$ measures the importance of t w.r.t. all of the documents in an object collection O. The text similarity of an object $o.d$ with respect to a user u is

$$\mathrm{TS}(o.d,u.d)=\sum_{t\in u.d\cap o.d} TF(t,o.d)\times IDF(t,O) \quad (2)$$

Fig. 1 Visibility measure



b. Spatial proximity: The spatial proximity of an object o w.r.t. a user u is measured using the minimum Euclidean distance, $\boldsymbol{d}^{\downarrow}(o.l, u.l)$ as:

$$\mathrm{SS}(o.l, u.l) = 1 - \frac{\boldsymbol{d}^{\downarrow}(o.l, u.l)}{d_{max}}, \tag{3}$$

where d_{max} is the maximum Euclidean distance between any two points in D. If the spatial location is any shape other than points, such as a line or rectangle, then the minimum Euclidean distance between those two shapes is used to compute the spatial proximity with the same equation.

c. Visibility measure: We now explain the visibility quantification for a line as the geometric shape of the spatial data. For each $o \in O$, $o.l$ is a line for the rest of the article for visibility, but the calculations are representative of any other geometric shape. An object o is considered visually relevant to a user u iff at least one point of $o.l$ is visible from the user $u.l$, i.e., there exists a point a over the line segment $o.l$ such that the straight line connecting a and $u.l$ does not pass through any other objects in O.

Previous work [7,24,42] has defined and used different metrics to quantify visibility. The metric, called "visual angle," used by [7] is the angle formed at the eye of a user by the extremities of an object viewed, which determines the perceived length of that object. We adopt this metric as the measure of visibility in this work. Specifically, the visibility measure in our work is computed as:

$$\overline{\mathrm{SS}}(o.l, u.l) = \frac{2arctan(\mathrm{VL}(o.l, u.l))}{180} \tag{4}$$

where the maximum possible visual angle is 180°, which is used to normalize the value of $\overline{\mathrm{SS}}(o.l, u.l)$ between [0,1], and $\mathrm{VL}(o.l, u.l)$ is the perceived length of o from $u.l$.

The perceived length of an object mainly depends on the distance and the viewing angle between the user and the object. If an object is viewed from an oblique angle, the perceived length of that object becomes smaller than the original length. The perceived length of o also decreases with the increase in the distance between o and the user u. As different parts of an object always have different distances and orientations w.r.t. a user, we use the *cumulative approach* presented by [42] to calculate visibility. Specifically, we divide the line $o.l$ into numerous infinitesimal segments of size at most ϵ such that, for each segment, $\Delta o.l$, the distances and the orientations of all points on $\Delta o.l$ w.r.t. to a user u can be considered as visually similar. For example, in Fig. 1, the segments are shown for an object o, and in Fig. 5, the segments are shown for a particular object o_2.

Let the straight line connecting the midpoint of a line segment $\Delta o.l$ and the point location of the user $u.l$ creates an angle $\angle(\Delta o.l, u.l)$ with $o.l$. Let the minimum distance of $\Delta o.l$ and the user $u.l$ be $\boldsymbol{d}^{\downarrow}(\Delta o.l, u.l)$. Then the perceived length of a segment $\Delta o.l$ w.r.t. $u.l$ is measured as:

$$\mathrm{VL}_{\Delta}(\Delta o.l, u.l) = \frac{\angle(\Delta o.l, u.l)}{90^{\circ}} \times \frac{len(\Delta o.l)}{\boldsymbol{d}^{\downarrow}(\Delta o.l, u.l)} \tag{5}$$

Here, if $\angle(\Delta o.l, u.l) = 90^{\circ}$, the perceived length of the line segment $\Delta o.l$ from a nominal distance is the same as its original length, $len(\Delta o.l)$. The perceived length of the entire $o.l$ w.r.t. $u.l$, $\mathrm{VL}(o.l, u.l)$, is obtained by summing up the perceived lengths of all the small segments $\Delta o.l$ that are visible from $u.l$:

$$\mathrm{VL}(o.l, u.l) = \sum_{\Delta o.l \text{ visible from } u.l} \mathrm{VL}_{\Delta}(\Delta o.l, u.l)$$

For example, in Fig. 5, to obtain the visibility of o_2, $\overline{\mathrm{SS}}(o_2.l, u.l)$, the perceived length of the segments of o_2 that are visible from u (represented with black) must be individually computed and summed as $\mathrm{VL}(o.l, u.l)$ in Eq. 4 to get the visibility value of object o_2 w.r.t. u.

Algorithm 1: BASELINE $(O, U, L, p, k,$ IR-tree$)$

```
1.1  B_p ← ∅
1.2  for each u ∈ U do
1.3  |  Traverse IR-tree to find kNN objects of u
1.4  |  R_k(u) ← similarity score of the kth ranked object of u
1.5  C ← Set of all combinations of ω keywords from W
1.6  for each ℓ ∈ L do
1.7  |  for each c ∈ C do
1.8  |  |  for each u ∈ U do
1.9  |  |  |  if c ∩ u.d ≠ ∅ then
1.10 |  |  |  |  p'.l = ℓ; p'.d = p ∪ c
1.11 |  |  |  |  if CS(p', u) > R_k(u) then B_p' ← u
1.12 |  |  if |B_p'| > |B_p| then
1.13 |  |  |  p.l ← ℓ; p.d ← p ∪ c
1.14 return p
```

3 Solution overview

One can think of a straightforward solution for the MaxST query consisting of the following steps shown in Algorithm 1: (i) find the top-k spatial–textual objects for all users in U individually using any of the existing techniques. Let the relevance score of the kth ranked object of a user $u \in U$

be $\mathcal{R}_k(u)$ (Lines 1.2–1.4). (ii) Generate all possible combinations C of ω keywords from W. (iii) For each candidate location $\ell \in L$, and each keyword combination $c \in C$, the total relevance score of ℓ and $p.d \cup c$ is computed for the users $u \in U$, where $c \cap u.d \neq \emptyset$. If this score is greater than $\mathcal{R}_k$, u is a RkNN of the object p with location ℓ and keyword set c. We track the $\langle \ell, c \rangle$ with the maximum number of RkNN and update the location and keywords of p accordingly (Lines 1.6–1.7). (iv) Finally, p is returned where the location and the keywords of p are the tuple $\langle \ell, c \rangle$ with the maximum number of RkNNs as the result.

Challenges: The straightforward method is computationally expensive for several reasons: (i) computing the top-k results for *all* users; (ii) iterating over *all* of the candidate locations; (iii) generating all combinations C of ω candidate keywords; and (iv) computing the relevance scores for all of the candidate location tuples, and $c \in C$ with respect to each user $u \in U$. We now show that the candidate selection part of the MaxST problem is NP-hard by reduction from the Maximum Coverage problem.

Lemma 1 *The MaxST problem is NP-hard.*

Proof Given a collection of sets $S = \{S_1, S_2, \ldots, S_n\}$, and a positive integer m, the Maximum Coverage (MC) problem is to find a subset $S' \subseteq S$ such that $|S'| \leq m$ and the number of covered elements by S', $|\cup_{S_i \in S'} S_i|$ is maximized. The MC problem is NP-hard [16].

Consider a special case of the MaxST problem where $\alpha = 1$. Here, the similarity score of the objects that contain at least one of the user keywords is measured by the spatial proximity using Eq. 1. Also assume that the number of candidate locations, $|L| = 1$ in this special case. So $p.l = \ell$, where ℓ is the only candidate location in L. For each candidate keyword $w \in W$, let $\mathcal{B}_w$ be the set of users that have p as a top-k object when w is included in $p.d$. So, a collection of the set of users exists, one for each $w \in W$. The goal of a MaxST query is to select at most ω keywords from W, such that the number of users for which p is a top-k object is maximum. That is, solving the maximum coverage problem is equal to finding a subset of the candidate keywords, $W' \subseteq W$, where $|W'| \leq \omega$ maximizes $|\cup_{w \in W'} \mathcal{B}_w|$.

Again, $\mathcal{B}_w$ for each $w \in W$ corresponds to each set S_i of the MC problem, where each user $u_j \in \mathcal{B}_w$ corresponds to the element of S_i. Therefore, finding a subset W' of the candidate keywords where $|W'| \leq \omega$ maximizing $|\cup_{w \in W'} \mathcal{B}_w|$ is equivalent to solving the maximum coverage problem. □

Therefore, scanning all combinations is not practical for a large number of candidates. To overcome these limitations, we seek techniques that:

- Efficiently compute the top-k objects for the users;
- Avoid computing the top-k objects for the users that cannot affect the result; and
- Prune the candidate locations and keywords that cannot be part of the final result.

A series of approaches to answer the MaxST query are presented. In each of these approaches, the idea is to avoid processing the candidates that cannot be a result, avoid computing the top-k objects of the users that do not have any effect on the final optimal result, share object retrieval costs among the users, and access the necessary objects only once. The methods differ in the pruning techniques that are applied to achieve these goals. In the following, an overview of three different solutions for MaxST queries is presented from a high level, with differences between each explained.

- GRP- TOPK approach: This approach consists of two separate modules to answer the MaxST problem. First, an efficient technique to compute the top-k objects for all users jointly is presented to address the first challenge mentioned above. Next, several pruning techniques are applied to discard the candidates that cannot be a result, where the score of the kth ranked object of the users is used to facilitate the pruning.
- INDIV- U approach: A limitation of the GRP- TOPK approach is that the top-k objects for all of the users must be computed. Therefore, the INDIV- U approach is proposed to avoid computing the top-k objects for users that do not affect the final result set. The idea is to estimate the number of objects that can have a higher similarity than a candidate for each user $u \in U$ with a single traversal of the objects. This number is used to prune the unpromising candidates and also the unnecessary users that cannot be an RkNN of any promising candidate.
- INDEX- U approach: In the previous approach, the users are pruned by checking each one individually against the candidates. So, a new index is proposed, the Modified IUR-tree (MIUR-tree) to store the users, where the motivation is that a hierarchical index structure over the set of users may exhibit a higher pruning capacity than checking them individually.

In Sects. 4, 5, 6 and 7, we propose approaches to answer a MaxST query, where the steps are described using the Euclidean distance as spatial similarity measure. In Sect. 8, extension of the solutions is presented to support answering a MaxST query in obstructed space, where the spatial relevance is measured as visibility.

4 Index structure

We propose a new index, the Min–Max IR-tree (MIR-tree), to index the set of objects O to support efficient processing of

the MaxST query. The MIR-tree is an extension of the IR-tree [9]. We first give an brief overview of the IR-tree, and then, we describe how we have extended the IR-tree to construct our proposed index.

An IR-tree is an R-tree [15] where each node is augmented with a reference to an inverted file [44] for the documents in the subtree. Each node R contains a number of entries, consisting of a reference to a child node of R, the MBR of all entries of the child node, and an identifier of a text description. If R is a leaf node, this is the identifier of the text description of an object. Otherwise, the text identifier is used for a pseudo-text description. The pseudo-text description is the union of all text descriptions in the entries of the child node. The weight of a term t in the pseudo-document is the maximum weight of the weights of this term in the documents contained in the subtree. Each node has a reference to an inverted file for the entries stored in the node. A posting list of a term t in the inverted file is a sequence of pairs $\langle d, \boldsymbol{w}(d,t)\rangle$, where d is the document id containing t, and $\boldsymbol{w}(d,t)$ is the weight of term t in d. In this article, the weight of a term t in a document d is computed using the TF·IDF metric described in Sect. 2.

4.1 Min–Max IR-tree (MIR-tree)

We propose the Min–Max IR-tree (MIR-tree) to index the objects. The objects are inserted in the same manner as in the IR-tree. However, unlike an IR-tree, each term is associated with both the maximum $\boldsymbol{w}^{\uparrow}(d,t)$ and minimum $\boldsymbol{w}^{\downarrow}(d,t)$ weights in each document. The posting list of a term t is a sequence of tuples $\langle d, \boldsymbol{w}^{\uparrow}(d,t), \boldsymbol{w}^{\downarrow}(d,t)\rangle$, where d is the document identifier containing t, $\boldsymbol{w}^{\uparrow}(d,t)$ is the maximum, and $\boldsymbol{w}^{\downarrow}(d,t)$ is the minimum weight of term t in d, respectively. If R is a leaf node, both weights are the same as the actual weight of the term t, $\boldsymbol{w}(d,t)$ in the IR-tree. If R is a non-leaf node, the pseudo-document of R is the union of all text descriptions in the entries of the child node. The maximum (minimum) weight of a term t in the pseudo-document is the maximum (minimum) weight in the union (intersection) of the documents contained in the subtree. If a term is not in the intersection, $\boldsymbol{w}^{\downarrow}(d,t)$ is set to 0.

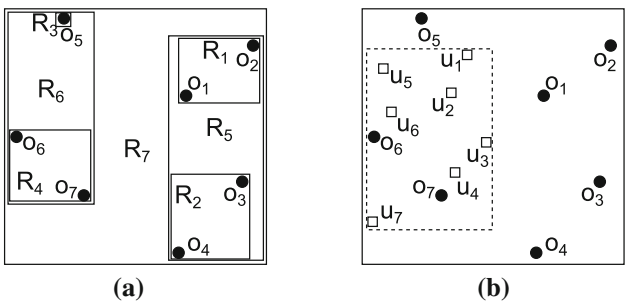


Fig. 2 The placement of the objects and the users. **a** Objects and MBRs, **b** objects and users

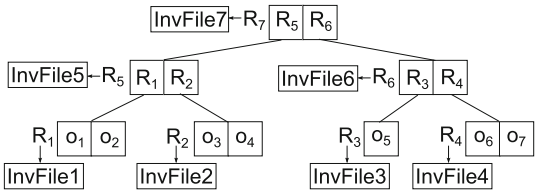


Fig. 3 Min–max IR-tree (MIR-tree)

Figure 2a shows the locations and the text descriptions of an example dataset $O = \{o_1, o_2, \ldots, o_7\}$, and Fig. 3 illustrates the MIR-tree for O. Table 2 presents the inverted files of the leaf nodes (InvFile 1 - 4) and the non-leaf nodes (InvFile 5 - 7) of the MIR-tree for the example objects in Fig. 2a. The tree structure of the MIR-tree is same as the IR-tree. As a specific example, the maximum (minimum) weight of term t_1 in entry R_4 of InvFile 6 is 2 (1), which is the maximum (minimum) weight of the term in the union (intersection) of documents (o_6, o_7) of the node R_4.

Index construction cost: In contrast to the IR-tree [9], the space requirements for the MIR-tree include an additional weight stored for the minimum text relevance for each term in each node. Specifically, for a node N, if the number of terms is M, the additional space is required to store $\sum_{i=1}^{M} n_i$ weights, where n_i is the number of objects in the posting list of term t_i in node N. The construction time of the MIR-tree is very similar to the original IR-tree. During tree construction, when determining the maximum weight of each term in a node, the minimum weight of that term can be determined concurrently. As the split and merge of the nodes are executed in the same manner as the IR-tree, the update costs of the MIR-tree are also same as that of the IR-tree.

The MIR-tree is an extension of the original IR-tree presented by [9], who also proposed other variants of the IR-tree, such as the DIR-tree, the CIR-tree, and the CDIR-tree, where both spatial and textual criteria are considered to construct the nodes of the tree. The same structures can be used during the construction of our proposed extension. For example, the nodes of the MIR-tree can be constructed in the same manner as the DIR-tree, and the posting lists of each node will contain both the minimum and maximum weights of the terms.

User grouping: Our goal is to access the necessary objects from disk and avoid duplicate retrieval of objects for different users. We form a group of users, denoted as a "super-user" (u^+), to facilitate the pruning of the objects and the candidates.

The "super-user" (u^+) is constructed such that $u^+.l$ is the MBR enclosing the locations of all users, $u^+.dUni$ is the union, and $u^+.dInt$ is the intersection of the keywords of all users, respectively. As an example, Fig. 2b shows the locations of the users $U = u_1, u_2, \ldots, u_7$ and the corresponding text descriptions are presented in Table 3. The location of the

Table 2 Posting lists for the example MIR-tree

Term	InvFile 1	InvFile 2	InvFile 3	InvFile 4	InvFile 5	InvFile 6	InvFile 7
t_1	$(o_1, 1, 1)$	$(o_3, 5, 5)$	$(o_5, 4, 4)$	$(o_6, 1, 1), (o_7, 2, 2)$	$(R_1, 1, 0), (R_2, 5, 0)$	$(R_3, 4, 4), (R_4, 2, 1)$	$(R_5, 5, 0), (R_6, 4, 1)$
t_2	$(o_1, 4, 4)$	–	$(o_5, 1, 1)$	–	$(R_1, 4, 0)$	$(R_3, 1, 1)$	$(R_5, 4, 0), (R_6, 1, 0)$
t_3	–	$(o_3, 5, 5)$	–	$(o_6, 1, 1)$	$(R_2, 5, 0)$	$(R_4, 1, 0)$	$(R_5, 5, 0), (R_6, 1, 0)$
t_4	$(o_2, 1, 1)$	$(o_4, 2, 2)$	–	$(o_7, 3, 3)$	$(R_1, 1, 0), (R_2, 2, 0)$	$(R_4, 3, 0)$	$(R_5, 2, 0), (R_6, 3, 0)$

Table 3 Text description of the users

Term	User						
	u_1	u_2	u_3	u_4	u_5	u_6	u_7
t_1	1	1	1	1	1	1	1
t_2	0	0	0	1	1	0	0
t_3	1	1	0	1	0	0	0
t_4	1	1	0	0	0	1	1

"super-user," $u^+.l$ is the MBR enclosing the locations of all the users, shown with a dotted rectangle. Here, the intersection of the keywords of all the users, $u^+.dInt$, is "t_1" and the union, $u^+.dUni$, is "t_1, t_2, t_3, t_4."

5 Grp-topk approach

In this approach, we assume that the top-k objects for all users are computed as a first step to answering a MaxST query. In our earlier work on this problem [8], an efficient technique to jointly process the top-k object computation for all users in U using a super-user was presented. Other approaches to efficiently batch process multiple top-k spatial–textual queries (users) could also be used [6]. Let the similarity score of the kth ranked object for each user $u \in U$, $\mathcal{R}_k(u)$, and the kth best lower-bound similarity score with respect to the super-user, $\mathcal{R}_k(u^+)$ such that for any user $u \in U$, $\mathcal{R}_k(u^+) \leq \mathcal{R}_k(u)$. As the details of an efficient solution to jointly compute the top-k objects for the users are available as previously described by [8], we proceed to present our method to efficiently find the best candidate combinations.

5.1 Candidate selection

As shown in Lemma 1, even when there is only a single candidate location, the candidate keyword selection process alone is NP-hard. Therefore, we propose a spatial-first pruning technique to select the best candidate combination of a location and a set of keywords.

For each candidate location ℓ, the idea is to estimate the number of users that can be in $\mathcal{B}_p$ if $p.l = \ell$ for the specific object p. Then the candidates are considered in a best-first manner so that the most promising candidates are processed first. Several pruning strategies are used during this process, which are described now.

Algorithm 2: SELECT _CANDIDATE(U, L, W, ω, k, p)

```
2.1  Initialize a max-priority queue QL.
2.2  B_p ← ∅.
2.3  for each ℓ ∈ L do
2.4      if CS↑(ℓ, u+) ≥ R_k(u+) then
2.5          for each u ∈ U do
2.6              if CS↑(ℓ, u) ≥ R_k(u) then
2.7                  B↑_ℓ ← u
2.8              if CS↓(ℓ, u) ≥ R_k(u) then
2.9                  B↓_ℓ ← u
2.10     ENQUEUE(QL, ℓ, |B↑_ℓ|)
2.11 while QL ≠ ∅ do
2.12     maxℓ ← DEQUEUE(QL)
2.13     if |B↑_maxℓ| < |B_p| then break else if |B↓_maxℓ| ≥ |B_p|
          then
2.14         p.l ← maxℓ
2.15     else
2.16         W' ← Find best candidate keyword set for maxℓ
             using approximate or exact method.
2.17         p'.l = maxℓ; p'.d = W'
2.18         if |B_p'| > |B_p| then
2.19             p.l ← maxℓ; p.d ← W'
2.20 return p
```

Algorithm 2 shows the pseudocode of the steps to select the candidate location and keywords for the MaxST problem. The pruning techniques used in this process use an upper- and a lower- bound estimation of relevance of the candidate combinations with respect to the users.

Upper-bound estimation: For each $\ell \in L$, the combined spatial–textual upper-bound similarity is computed in two steps: (i) for the super-user u^+, denoted as $\text{CS}^\uparrow(\ell, u^+)$ such that for any user $u \in U$, the similarity between p and u is at most the $\text{CS}^\uparrow(\ell, u^+)$, when $p.l = \ell$; and (ii) for each individual $u \in U$ such that the similarity $\text{CS}(p, u)$ is at most $\text{CS}^\uparrow(\ell, u)$, when $p.l = \ell$.

When using Euclidean distance for spatial similarity, the spatial upper-bound $\text{SS}^\uparrow(\ell, u^+)$ is computed from Eq. 3 using the minimum Euclidean distance between ℓ and u^+, as $u^+.l$ is the MBR for all of the users. For text relevance, a straightforward way is to consider the relevance as 1 (maximum), when the score is normalized within [0, 1]. But we can achieve a tighter bound using the following lemma:

Lemma 2 *Let $W^{\uparrow}$ be the set of ω number of keywords of the highest weights from $(u^{+}.dUni \cap W)$. The text relevance between $p.d$ and a user $u \in U$ after adding at most ω number of candidate keywords is always less than or equal to the score $TS((p.d \cup W^{\uparrow}), u^{+}.dUni)$,*

$$TS^{\uparrow}(p, u^{+}) = TS((p.d \cup W^{\uparrow}), u^{+}.dUni)\,.$$

Proof The text relevance between a user u and p can change by adding only the keywords that are present in $u.d$. As $u^{+}.dUni$ is the union of the text of all the users in U, the text relevance w.r.t. any user u can be increased only by adding the candidate keywords that are present in $u^{+}.dUni$. Let w_1 and w_2 be two keywords in $W^{\uparrow}$ where the weight of w_1 is greater than the weight of w_2 and $\omega = 1$. If a user u has both w_1 and w_2 in the text description, then from Eq. 2, the text relevance of p w.r.t. u obtained by adding w_1 must be equal to or greater than the text relevance obtained by adding w_2. Even if a user u does not have all the keywords of $W^{\uparrow}$ in $u.d$, the lemma still provides an upper-bound estimation of text relevance that can be achieved by adding ω number of candidate keywords. □

So, the upper-bound estimation of relevance of a candidate location w.r.t. the super-user u^{+} is

$$\mathrm{CS}^{\uparrow}(\ell, u^{+}) = \alpha \cdot \mathrm{SS}^{\uparrow}(\ell, u^{+}.l) + (1-\alpha) \cdot \mathrm{TS}^{\uparrow}(p, u^{+})\,.$$

Similarly, an upper-bound estimation of a candidate location ℓ w.r.t. any particular user u can be computed as

$$\mathrm{CS}^{\uparrow}(\ell, u) = \alpha \cdot \mathrm{SS}^{\uparrow}(\ell, u.l) + (1-\alpha) \cdot \mathrm{TS}^{\uparrow}(p, u)\,.$$

Here, $\mathrm{TS}^{\uparrow}(p.d, u.d) = \mathrm{TS}(p.d \cup {W_u}^{\uparrow}, u.d)$, where ${W_u}^{\uparrow}$ is the set of ω number of keywords of the highest weights from $(u.d \cap W)$.

Lower-bound estimation: For text relevance, the minimum score is computed from the original text description of p. The spatial lower bound is computed using the maximum Euclidean distance. So, the lower-bound estimation of $\ell \in L$ w.r.t. u^{+} is:

$$\mathrm{CS}^{\downarrow}(\ell, u^{+}) = \alpha \cdot \mathrm{SS}^{\downarrow}(\ell, u^{+}.l) + (1-\alpha) \cdot \mathrm{TS}^{\downarrow}(p.d, u^{+}.dInt)\,.$$

Pruning techniques: We denote the set of users that can be in $\mathcal{B}_p$ for $p.l = \ell$ as $\mathcal{B}^{\uparrow}{}_{\ell}$, and the set of users that are definitely in $\mathcal{B}_p$ when $p.l = \ell$ as $\mathcal{B}^{\downarrow}{}_{\ell}$. The number of users that find p as a top-k object is initialized as an empty set. The steps and the pruning strategies employed in Algorithm 2 can be summarized as follows:

- Initialize necessary user lists: As the similarity score of the kth ranked object for any user u, $\mathcal{R}_k(u)$, satisfies the condition $\mathcal{R}_k(u^{+}) \leq \mathcal{R}_k(u)$. Therefore, if $\mathrm{CS}^{\uparrow}(\ell, u^{+}) < \mathcal{R}_k(u^{+})$, then no user can have p as a top-k object for the candidate location ℓ. Otherwise, $\mathrm{CS}^{\uparrow}(\ell, u)$ is computed for each user. If $\mathrm{CS}^{\uparrow}(\ell, u) \geq \mathcal{R}_k(u)$, then u is included in $\mathcal{B}^{\uparrow}{}_{\ell}$. A list of such users, $\mathcal{B}^{\uparrow}{}_{\ell}$, is obtained for each candidate location ℓ, (Lines 2.6–2.7). For each ℓ, if the lower-bound similarity $\mathrm{CS}^{\downarrow}(\ell, u) \geq \mathcal{R}_k(u)$, then u is added to the corresponding list $\mathcal{B}^{\downarrow}{}_{\ell}$ (Lines 2.8–2.9).
- Here, a *best-first traversal* technique is exploited. A max-priority queue QL of candidate locations is maintained according to the cardinality $|\mathcal{B}^{\uparrow}{}_{\ell}|$, so that the most promising candidates are processed first. In each iteration, the location, $max\ell$, with the maximum $|\mathcal{B}^{\uparrow}{}_{\ell}|$ is selected (Line 2.12).
- As the set $\mathcal{B}^{\uparrow}{}_{\ell}$ for the candidates are maintained based on an upper bound, the cardinality of $\mathcal{B}^{\uparrow}{}_{max\ell}$ is less than the best $|\mathcal{B}_p|$ found so far. So, a better tuple from the subsequent entries of QL is not possible. Thus, the computation can be *early terminated* (Line 2.13).
- Since all users in $\mathcal{B}^{\downarrow}{}_{\ell}$ have p as a top-k object for $p.l = max\ell$, irrespective of the keyword selection, a check to see whether $|\mathcal{B}^{\downarrow}{}_{max\ell}|$ is greater than the current best $|\mathcal{B}_p|$ can be used to *avoid computing the candidate keywords* for this condition (Lines 2.14–2.15).
- Otherwise, the best candidate keyword set, W', is determined for $max\ell$. An approximate or an exact method presented in the following section is used to select W' (Line 2.17). p is updated with $max\ell$ and W' accordingly (Lines 2.19–2.20).

5.2 Candidate keyword selection

Recall that the best candidate keyword W' set that provides the maximum cardinality of $\mathcal{B}_p$ has to be determined for $p.l = max\ell$ (Line 2.17) in Algorithm 2. As this is an NP-hard problem, an approximation algorithm is first developed. An exact method that uses several pruning strategies is also presented, which can serve as a naive baseline.

5.2.1 Approximate algorithm

The candidate keyword selection problem was shown to be NP-hard in Lemma 1 using a reduction from the Maximum Coverage (MC) problem. A greedy algorithm with the best-possible polynomial time exists with $(1 - 1/e) \simeq 0.632$ approximation ratio for the MC problem [10]. Inspired by this algorithm, we propose an approximate algorithm to select the candidate keywords in our algorithm when $p.l = max\ell$ (Line 2.17 of Algorithm 2). However, the assumption of the solution in [10] is that the objective function needs to be submodular. As the objective function of MaxST problem, which

Algorithm 3: APPROXIMATE $(\mathcal{B}_{max\ell}, W, \omega, k)$

3.1 $p'.l = max\ell$; $W' \leftarrow \varnothing$; $CU \leftarrow \varnothing$
3.2 **for** *each* $w \in W$ **do**
3.3 | **for** *each* $u \in \mathcal{B}_{max\ell}$ **do**
3.4 | | $W^{\uparrow}{}_{w,u} \leftarrow$ set of ω highest weighed keywords from $W \cap u.d$ and $w \cap W^{\uparrow}{}_{w,u} \neq \varnothing$
3.5 | | $p'.d = W^{\uparrow}{}_{w,u}$
3.6 | | **if** $CS(p', u) > \mathcal{R}_k(u)$ **then**
3.7 | | | $\mathrm{LW}_w \leftarrow u$
3.8 **while** $|W'| \leq \omega$ **do**
3.9 | $w \leftarrow$ the keyword from W with the maximum $|\mathrm{LW}_w|$
3.10 | $W' \leftarrow W' \cup w$
3.11 | $CU \leftarrow CU \cup \mathrm{LW}_w$
3.12 | $W \leftarrow W - w$
3.13 | **for** *each* $w \in W$ **do**
3.14 | | $\mathrm{LW}_w \leftarrow \mathrm{LW}_w - CU$
3.15 return W'

Algorithm 4: EXACT$(U, max\ell, \mathcal{B}^{\uparrow}{}_{max\ell}, W, \omega, k)$

4.1 $\mathrm{WU} \leftarrow \bigcup_{u \in \mathcal{B}^{\uparrow}{}_{max\ell}} (u.d)$
4.2 $W' \leftarrow \varnothing$; $best \leftarrow 0$
4.3 **if** $|W \cap WU| \leq \omega$ **then**
4.4 | $W' \leftarrow (W \cap \mathrm{WU})$
4.5 **else**
4.6 | $C \leftarrow$ combinations of ω number of keywords from $W \cap \mathrm{WU}$.
4.7 | $p'.l = max\ell$
4.8 | **for** *each* $c \in C$ **do**
4.9 | | $p'.d = c$
4.10 | | **for** *each* $u \in \mathcal{B}^{\uparrow}{}_{\max\ell}$ **do**
4.11 | | | **if** $CS^{\downarrow}(\max\ell, u) \geq \mathcal{R}_k(u)$ **then**
4.12 | | | | $\mathcal{B}_{p'} \leftarrow u$
4.13 | | | **else if** $c \cap u.d \neq \emptyset$ **then**
4.14 | | | | **if** $CS(p', u) \geq \mathcal{R}_k(u)$ **then** $\mathcal{B}_{p'} \leftarrow u$
4.15 | | **if** $|\mathcal{B}_{p'}| > best$ **then**
4.16 | | | $W' \leftarrow c$; $best \leftarrow |\mathcal{B}_{p'}|$
4.17 return W'

is to maximize the number of RkNNs, is non-submodular, the approximation ratio of that solution does not hold. We present the steps of the approximate approach in Algorithm 3 where some preprocessing is required.

Preprocessing: For each $w \in W$, we generate a list LW_w of the users such that these users can be in $\mathcal{B}_p$ based on an upper-bound estimation, where $p.d = W'$ and $W' \cap w \neq \varnothing$. As the $\mathcal{B}^{\uparrow}{}_{max\ell}$ is already computed based on this upper bound, only the users in $\mathcal{B}^{\uparrow}{}_{max\ell}$ need to be considered for this step. Let $W^{\uparrow}{}_{w,u}$ be a set of the ω highest weighed keywords from $W \cap u.d$ such that $W^{\uparrow}{}_{w,u} \cap w \neq \varnothing$. When $p.d = W^{\uparrow}{}_{w,u}$ and $p.l = max\ell$, a user u can be in $\mathcal{B}_p$ if $\mathrm{CS}(p, u) \geq \mathcal{R}_k(u)$. Such users are included in the corresponding list, LW_w for each $w \in W$ (Lines 3.2–3.3).

Approximating the best candidate keyword set: Recall that in the MC problem, the objective is to find a subset $S' \subseteq S$ such that $|S'| \leq m$ and the number of covered elements by S', $|\cup_{S_i \in S'} S_i|$ is maximized. In our case, the collection of the sets are the collection of LW_w for each w and m is ω. The greedy approach of MC is applied in our problem to find the best set of candidate keywords W' of size ω such that $|\cup_{w \in W'} \mathrm{LW}_w|$ is maximized. This set W' is returned as the best candidate keyword set for the location $max\ell$ (Lines 3.8–3.15).

The approximate approach greedily selects the keyword with the highest number of uncovered RkNNs in each iteration, until ω keywords are selected. Thus, the iteration execution number of the approximate algorithm is ω.

5.2.2 Exact algorithm

The number of candidates can be small in some applications. Moreover, the search space can be pruned using several strategies when selecting the candidate keyword set. This motivates us to develop an exact algorithm for selecting the best keyword set W' of the MaxST query. The pseudocode is presented in Algorithm 4, and the pruning techniques are now explained.

- Pruning users: According to the definition of $\mathrm{CS}^{\uparrow}(\ell, u)$, only the users in $\mathcal{B}^{\uparrow}{}_{max\ell}$ can have p as a top-k object when $p.l = max\ell$. So only the users in $\mathcal{B}^{\uparrow}{}_{max\ell}$ must be considered.
- Pruning candidate keywords: Let the union of the text description of the users in $\mathcal{B}^{\uparrow}{}_{max\ell}$ be WU (Line 4.1). Only the candidate keywords that are contained in at least one of those users, $W \cap \mathrm{WU}$, are necessary.
- Let C be the set of the combinations of ω number of keywords from $W \cap \mathrm{WU}$. For a keyword combination $c \in C$, only those users where $c \cap u.d \neq \varnothing$ are processed.
- Early termination: If $|W \cap \mathrm{WU}| \leq \omega$, this is the only possible candidate keyword set. So the process terminates and $W \cap \mathrm{WU}$ is returned as the best candidate keyword set for $max\ell$ as shown in Lines 4.3–4.4.
- If the lower-bound relevance, $\mathrm{CS}^{\downarrow}(max\ell, u) \geq \mathcal{R}_k(u)$, then u is included in $\mathcal{B}_{p'}$, where $p'.l = max\ell$ and $p'.d$ is the candidate keyword combination c currently under consideration (Lines 4.11–4.12). If the cardinality of $\mathcal{B}_{p'}$ is greater than that of the current best keyword combination, the current best is updated (Lines 4.15–4.16).

6 Indiv-U approach

Instead of computing the top-k objects of each of the users as the GRP- TOPK approach, the idea of the INDIV- U approach is to estimate the number of objects with a higher similarity than each candidate w.r.t. each $u \in U$ with a single traversal on the objects. This number is used to prune the

users and the candidates that cannot affect the result. As the steps of the algorithm rely on computing the similarity estimations of the objects w.r.t. the users, we first present the similarity bounds, and then, we present the steps of our algorithm.

6.1 Similarity bounds of an MIR-tree node of objects

We use each node of the MIR-tree to estimate the similarity of the objects stored in that node. Here we present an upper and a lower bound of similarity estimation of a node E of the MIR-tree w.r.t. (i) the super-user and (ii) an individual user u. These bounds are then used to facilitate the pruning of the objects nodes and pruning of the users from consideration. The maximum spatial–textual similarity between any node E of the MIR-tree and the super-user u^+ is computed as

$$\mathrm{CS}^{\uparrow}(E, u^+) = \alpha \cdot \mathrm{SS}^{\uparrow}(E.l, u^+.l) + (1-\alpha) \cdot \mathrm{TS}^{\uparrow}(E.d, u^+.dUni)\,,$$

where $\mathrm{SS}^{\uparrow}$ is the maximum spatial similarity computed from the minimum Euclidean distance between the two MBRs using Eq. 3, and $\mathrm{TS}^{\uparrow}$ is the maximum textual similarity between $E.d$ and the union of the keywords of the users, and $u^+.dUni$ is computed by

$$\mathrm{TS}^{\uparrow}(E.d, u^+.dUni) = \sum_{t \in u^+.dUni \cap E.d} \boldsymbol{w}^{\uparrow}(E.d, t)\,,$$

where $\boldsymbol{w}^{\uparrow}(E.d, t)$ is the maximum weight of the term t in the associated document of node E. As described in Sect. 4, if E is a non-leaf node, $\boldsymbol{w}^{\uparrow}(E.d, t)$ is the maximum weight in the union of the documents contained in the subtree of E. Otherwise, $\boldsymbol{w}^{\uparrow}(E.d, t)$ is the weight of term t in document $E.d$ computed using Eq. 2.

We now present a lemma that enables us to estimate an upper bound on the relevance between any user $u \in U$, and any object node E using the super-user u^+, where E is a node of the MIR-tree.

Lemma 3 *$\forall u \in U$, $CS^{\uparrow}(E, u^+)$ is an upper-bound estimation of $CS(E, u)$. For any object node E, $CS(E, u) \leq CS^{\uparrow}(E, u^+)$.*

Proof Recall that the $u^+.l$ of u^+ is the MBR of the locations for all of the users in U. For an object node E in the MIR-tree, $\mathrm{SS}^{\uparrow}(E, u^+)$ is computed from the minimum Euclidean distance between the two MBRs of E and $u^+.l$. As the location $u.l$ of any user $u \in U$ is inside the rectangle $u^+.l$, the value $\mathrm{SS}(E, u)$ must be less than or equal to $\mathrm{SS}^{\uparrow}(E, u^+)$. For textual similarity, as $u^+.dUni = \cup_{u \in U} u.d$, the maximum textual similarity score between any user $u \in U$, and any object node E that can be achieved is $\mathrm{TS}^{\uparrow}(E, u^+)$ from Eq. 2. Since the spatial–textual score $\mathrm{CS}(E, u)$ is the weighted sum of the corresponding spatial and textual scores, $\forall u \in U$, the score $\mathrm{CS}(E, u)$ must also be less than or equal to $\mathrm{CS}^{\uparrow}(E, u^+)$. □

Algorithm 5: INDIV- U $(O, U, L, p, k,$ MIR-tree$)$

```
5.1  Initialize max-priority queue QL, QE.
5.2  Initialize a min-priority queue LS.
5.3  Initialize an array LO_u of min-priority queues for each u ∈ U.
5.4  LU ← U
5.5  for each ℓ ∈ L do
5.6      B↓_ℓ ← ∅; B↑_ℓ ← U
5.7  ⟨ℓ, W'⟩_best ← ∅; E ← MIR-tree(root)
5.8  ENQUEUE(QE, E, CS↓(E, u+))
5.9  while QE ≠ ∅ do
5.10     E ← DEQUEUE(QE)
5.11     if E is an object then
5.12         if |LS| < k || CS↑(E, u+) ≥ R_k(u+) then
5.13             ENQUEUE(LS, E, CS↓(E, u+))
5.14             Update R_k(u+)
5.15             for each u ∈ LU do
5.16                 next ← true
5.17                 if |LO_u| < k || CS↑(E, u) ≥ R_k(u) then
5.18                     ENQUEUE(LS, E, CS↓(E, u))
5.19                     Update R_k(u)
5.20                     for each ℓ ∈ L do
5.21                         if u is in B↑_ℓ & |LO_u| ≥ k &
                             CS↑(ℓ, u) < R_k(u) then
5.22                             Remove u from B↑_ℓ
5.23                         else next ← false
5.24                     if next is true then
5.25                         Remove u from LU
5.26     else if |LS| < k || CS↑(E, u+) ≥ R_k(u+) then
5.27         for each element e of E do
5.28             for each u ∈ LU do
5.29                 if |LO_u| < k || CS↑(E, u) ≥ R_k(u) then
5.30                     ENQUEUE(QE, e, CS↓(e, u+))
5.31                     break
5.32 for each ℓ ∈ L do
5.33     for each u ∈ B↑_ℓ do
5.34         if CS↓(ℓ, u) ≥ R_k(u) then B↓_ℓ ← u
5.35     ENQUEUE(QL, ℓ, |B↑_ℓ|)
5.36 Execute Lines 2.11- 2.21 of Algorithm 2.
```

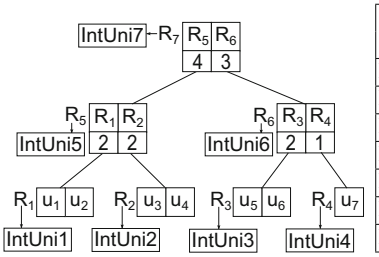
Lemma 3 shows that $\mathrm{CS}^{\uparrow}(E, u^+)$ is a correct upper-bound estimation for relevance between a node E of the MIR-tree, and any $u \in U$. Similarly, a lower-bound relevance can be computed as: $\mathrm{CS}^{\downarrow}(E, u^+) = \alpha \cdot \mathrm{SS}^{\downarrow}(E.l, u^+.l) + (1-\alpha) \cdot \mathrm{TS}^{\downarrow}(E.d, u^+.dInt)$, where $\mathrm{SS}^{\downarrow}$ is computed from the maximum Euclidean distance between the two MBRs, $\mathrm{TS}^{\downarrow}$ is the minimum textual relevance between E and $u^+.dInt$ is computed using the minimum weights of the terms in E. Similar to the upper-bound estimation, for the lower bound, the property $\forall u \in U, \mathrm{CS}(E, u) \geq \mathrm{CS}^{\downarrow}(E, u^+)$ always holds.

6.2 Algorithm

We present our approach that avoids computing the top-k object for users that do not affect the final result. Here, we apply a spatial-first strategy as well. Algorithm 5 presents the pseudocode of the approach. Recall that the set of users that can be in $\mathcal{B}_p$ for $p.l = \ell$ is denoted as $\mathcal{B}^{\uparrow_\ell}$, and the set of users that are definitely in $\mathcal{B}_p$ when $p.l = \ell$ as $\mathcal{B}^{\downarrow_\ell}$. The algorithm works as follows:

- Initialization: For a candidate location $\ell \in L$, $\mathcal{B}^{\downarrow_\ell}$ is initialized with an empty set and $\mathcal{B}^{\uparrow_\ell}$ is initialized with the set U (Line 5.6). Let LU be the set of the users that are in $\mathcal{B}^{\uparrow_\ell}$ of at least one candidate $\ell \in L$, which is also initialized with all the users in U. For each user u, a priority queue LO_u is maintained to track the current top-k objects of u found so far. A separate priority queue LS is also maintained for the super-user to track k objects with the best lower-bound similarity found so far.
- Lines 5.11–5.19 show how LS and LO_u for each user u are filled with k objects with the highest lower-bound similarity values found so far. The actual objects are used instead of object nodes in LS and each LO_u for better relevance estimations. The corresponding values of $\mathcal{R}_k(u)$ and $\mathcal{R}_k(u^+)$ are also updated.
- Pruning object nodes using the super-user: If the upper-bound similarity $\text{CS}^{\uparrow}(E, u^+)$ is less than the current $\mathcal{R}_k(u^+)$, then E cannot contain any object that can be a top-k object for any users in U (Line 5.26). Otherwise, the upper-bound similarity $\text{CS}^{\uparrow}(E, u)$ is computed w.r.t. each user $u \in \text{LU}$.
- Pruning object nodes for individual user: Similarly, if $\text{CS}^{\uparrow}(E, u)$ is less than $\mathcal{R}_k(u)$, then E for u does not need to be considered. If E is not needed for any of the users currently in LU, the subtree rooted at E can be pruned from further consideration (Lines 5.27–5.29).
- Pruning users for candidate locations: If the upper-bound similarity $\text{CS}^{\uparrow}(\ell, u)$ of a candidate ℓ w.r.t. a user u is less than the current $\mathcal{R}_k(u)$, then u cannot be in $\mathcal{B}^{\uparrow_\ell}$ for $p.l = \ell$. If u is discarded from the $\mathcal{B}^{\uparrow_\ell}$ for all of the candidates, then u is removed from LU as well. Computing the top-k objects for such users can be avoided (Lines 5.20–5.25).
- The set of users that are definitely an RkNN for $p.l = \ell$, $\mathcal{B}^{\downarrow_\ell}$, are found using their corresponding $\mathcal{R}_k(u)$ and $\text{CS}^{\downarrow}(\ell, u)$ values (Lines 5.32–5.34).

If a node E cannot be pruned, the entries of E are retrieved and placed in the queue. As traversal down the tree continues, the similarity bounds become closer to the actual values. When the leaf nodes are reached, the values for the actual objects are computed instead of nodes that cannot be pruned, and the objects that can be a top-k object of the necessary users are found. After the traversal of the MIR-tree, the list of users is obtained, $\mathcal{B}^{\uparrow_\ell}$ and $\mathcal{B}^{\downarrow_\ell}$ for each candidate location, and then, the rest of the candidate selection process is the same as the GRP- TOPK approach (Lines 2.11–2.21 of Algorithm 2).



ID	Int. ($t_1t_2t_3t_4$)	Union ($t_1t_2t_3t_4$)
IntUni1	1 0 1 1	1 0 1 1
IntUni2	1 0 0 0	1 1 1 0
IntUni3	1 0 0 0	1 1 0 1
IntUni4	1 0 0 1	1 0 0 1
IntUni5	1 0 0 0	1 1 1 1
IntUni6	1 0 0 0	1 1 0 1
IntUni7	1 0 0 0	1 1 1 1

Fig. 4 Example of modified IUR-tree (MIUR-tree)

Algorithm 6: INDEX- U $(O, L, p, k,$ MIR-tree, MIUR-tree$)$

```
6.1  Initialize a max-priority queue QL.
6.2  EU ← MIUR-tree(root)
6.3  Compute R_k(EU).
6.4  RO_EU ← List of objects o with CS↑(o, EU) ≥ R_k(EU) (using
     Algorithm 1 of Choudhury et al. [8])
6.5  for each ℓ ∈ L do
6.6      if CS↑(ℓ, EU) ≥ R_k(EU) then B↑_ℓ ← EU
         ENQUEUE(QL, ℓ, |B↑_ℓ|)
6.7  while QL ≠ ∅ do
6.8      maxℓ ← DEQUEUE(QL)
6.9      if B↑_ℓ ≠ ∅ then
6.10         EU ← node in B↑_ℓ with maximum users
6.11         for each eu ∈ EU do
6.12             Update R_k(eu) by executing Lines 2.3 - 2.11 of
                 Algorithm 2 by Choudhury et al. [8]
6.13             Update RO_eu
6.14             for each ℓ ∈ L do
6.15                 if B↑_ℓ contains EU || CS↑(ℓ, eu) ≥ R_k(eu)
                     then B↑_ℓ ← eu
6.16         Update QL
6.17     else
6.18         Execute Lines 5.34- 5.36 of Algorithm 5.
```

7 Index-U approach

In the INDIV- U approach presented in the previous section, the users are pruned by checking them individually against the candidates and the retrieved objects. In this section, we propose a new index, the Modified IUR-tree (MIUR-tree) to store the users, where the motivation is to improve efficiency by applying the pruning techniques over a hierarchical structure of the users instead of the individual users.

Modified IUR-tree (MIUR-tree): An MIUR-tree is essentially an R-tree where each node is augmented with a reference to the union and the intersection vector of the keywords appearing in the subtree. Each node R contains a number of entries, each consists of a reference to a child node, the MBR of all entries of the child node, and an identifier of a vector of key-

words. If R is a leaf node, this is the identifier of the vector of the text description of an object o. Otherwise, it has a reference to the union and intersection of all text descriptions in the entries of the child node. It also contains the number of actual objects stored in the subtree rooted at R.

Figure 4 illustrates the MIUR-tree for $U = \{u_1, u_2, \ldots, u_7\}$ of Fig. 2b, where the MBRs are constructed according to the IR-tree (not shown in the figure), and the table shows the text vectors of the nodes for the users presented in Table 3.

Algorithm: The pseudocode of the approach is presented in Algorithm 6. The root of the MIUR-tree is essentially the same as the super-user u^+ in the previous methods. The MIR-tree of the objects is traversed for the root node, EU of the MIUR-tree, to obtain the kth best lower bound $\mathcal{R}_k(EU)$ and the list RO of the object, such that each $o \in RO, \text{CS}^{\uparrow}(o, EU) \geq \mathcal{R}_k(EU)$ (Lines 6.3–6.4). The details of the traversal are explained in Algorithm 1 of our previous publication [8].

For each $\ell \in L$, a list $\mathcal{B}^{\uparrow}{}_{\ell}$ is maintained, but unlike Algorithm 2, $\mathcal{B}^{\uparrow}{}_{\ell}$ may now contain user nodes. In each iteration, the location $max\ell$ is selected with the maximum $|\mathcal{B}^{\uparrow}{}_{\ell}|$. If there is a user node in a $\mathcal{B}^{\uparrow}{}_{\ell}$, the number of actual users stored in that subtree is used to compute the number of users in $\mathcal{B}^{\uparrow}{}_{\ell}$. The following steps are executed to access the MIUR-tree-

1. If there is any non-leaf node in $\mathcal{B}^{\uparrow}{}_{max\ell}$, the non-leaf node $EU \in \mathcal{B}^{\uparrow}{}_{max\ell}$ is dequeued with the maximum number of users stored in the subtree.
   (a) For each element $eu \in EU$, Algorithm 2 in the work [8] is executed to update $\mathcal{R}_k(eu)$ using the list of objects $RO(EU)$ of its parent node. The list $RO(eu)$ is also updated (Lines 6.13–6.14).
   (b) For each $\ell \in L$ including $max\ell$, if $EU \in \mathcal{B}^{\uparrow}{}_{\ell}$, $\mathcal{B}^{\uparrow}{}_{\ell}$ is updated with the users $eu \in EU$ based on the corresponding upper-bound scores. The priority queue QL is also updated. In this way, a node of the MIUR-tree must only be accessed at most once.
2. Otherwise, the rest of the algorithm to find the best candidate location and keyword set combination of the MaxST query is same as Lines 5.34–5.36 of Algorithm 5.

In this best-first method, the users that are in the list $\mathcal{B}^{\uparrow}{}_{\ell}$ of the most promising candidates are accessed first. In addition, the top-k objects are not computed for the users that are not necessary to determine the result candidate combinations.

8 Adding visibility requirements

In this section, we extend our solutions to support answering the MaxST query in obstructed space, where the spatial relevance of an object is its visibility w.r.t. a user.

Challenges with visibility measures: The additional challenge of the visibility measure is that the visibility of an object o w.r.t. a user u depends on their locations, and the locations of the other objects (obstacles) in between them, where the spatial proximity (e.g., Euclidean distance) of o w.r.t. u depends only on their own locations. Therefore, to incorporate the visibility measure in the solutions presented in Sects. 5, 6, and 7, we seek techniques that:

- Efficiently estimate and calculate the visibility of an object or a candidate without requiring the retrieval of other objects whenever possible.
- Pruning of unnecessary objects, candidates, and users based on visibility.

Visibility extension overview: At the highest level, the following modifications are made in the solutions of Sects. 5, 6, and 7 to incorporate visibility and achieve the above goals:

1. The MIR-tree is extended, denoted as an *OIR-tree*, to store the set of objects O, along with some additional information to support visibility computation. Specifically, an auxiliary Quadtree structure is maintained to identify the portions of space that are visible to a user and associate that information with the OIR-tree. A space partitioning technique is used to construct the auxiliary Quadtree to facilitate finding the visible segments of an object or a candidate location.
2. As our proposed solutions rely on estimating the similarity bounds, the visibility estimation bounds of a candidate and an object o w.r.t. a user (and super-user) using the nodes of the OIR-tree are presented. These bounds do not require the retrieval of the other objects in between o and the user to estimate visibility, which is a major contribution of this work.
3. Several additional pruning techniques are also presented which use visibility to prune unnecessary objects, users, and candidates.

In Sect. 8.2, the extension of our proposed indexes to incorporate the visibility measure is presented. The modifications of the algorithms to answer the MaxST query for visibility are explored in Sect. 8.3.

8.1 Preliminaries

In this section, the notion of an *obstructed region* of a user is presented, which is crucial to measure the visibility.

Definition 2 Obstructed region ($OR(\mathbf{u})$). Given a set of obstacles O in space, an obstructed region w.r.t. a user u, $OR(u)$ consists of all the points in the space such that a

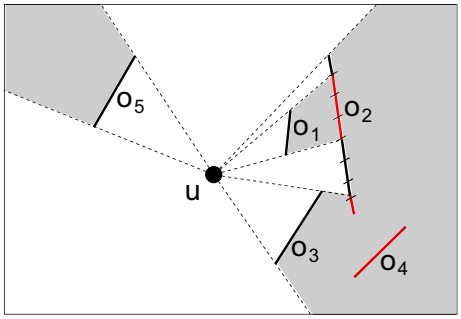


Fig. 5 Obstructed region

straight line connecting any of these points and $u.l$ passes through at least one object $o \in O$.

Based on the definition, an important observation is: If an object is completely inside the obstructed region of a user u, no point of that object is visible from u. As shown in Fig. 5, the shaded region is the obstructed region of the user u due to the presence of the objects $o_1, \ldots, o_5$. The visible portions of the objects are represented by black, and the obstructed portions (the portions that are inside the obstructed region) are represented with red. As the object o_4 is completely inside the obstructed region of u, o_4 is not visible at all from u.

8.2 Index structure

First we present the construction of the auxiliary Quadtree and then the details of the modifications to the MIR-tree to construct a OIR-tree.

8.2.1 Auxiliary Quadtree

The purpose of this auxiliary structure is to quickly find which part of an object or a candidate is obstructed from a user. To achieve this goal, the obstructed region $OR(u)$ for each user $u \in U$ is first constructed. The space is then partitioned using a Quadtree. For each cell ζ of the Quadtree, a list of users, $OL(\zeta)$, is maintained based on the visibility of ζ from those users. The idea is that the visibility of an object o w.r.t. any user can be estimated from the cells with which $o.l$ intersects. Moreover, if an object is completely inside cells that are obstructed from a user u, that object can be safely discarded from consideration for u.

A top-down approach is used to populate the list of users $OL(\zeta)$ associated with each cell ζ of the Quadtree. Specifically, $OL(\zeta)$ consists of the ID of the users, such that for each user $u \in OL(\zeta)$, ζ is completely inside the obstructed region $OR(u)$. Starting from the root, such users are found and added in the list $OL(\zeta)$ of the corresponding cell ζ. If a user u is included in $OL(\zeta)$ of a cell ζ, it implies that all the descendent cells of ζ are also contained inside $OR(u)$. Therefore, repeatedly storing u for the descendants of that cell ζ is not required. If an additional user u' is found for which a cell ζ is completely inside $OR(u')$, but u' is not included in any OL of the ancestor cells of ζ, only then is u' included in $OL(\zeta)$. This is illustrated with the example shown in Figs. 6 and 7.

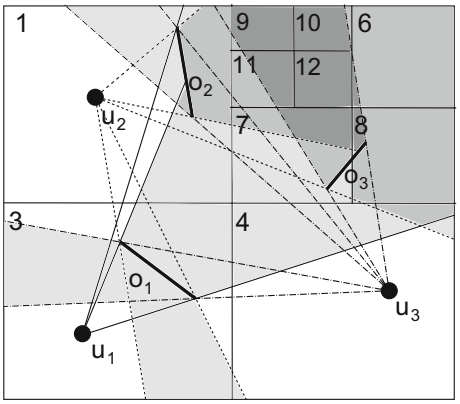
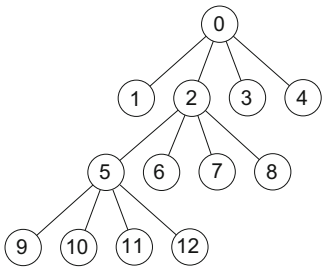


Fig. 6 Auxiliary Quadtree construction



Cell ID	OL(c)
c_2	u_1
c_5	u_2
c_6	u_2
c_{10}	u_3
c_{12}	u_3

Fig. 7 Auxiliary Quadtree partitioning and user list

Example 3 Let the space be partitioned using a Quadtree into 10 cells as in Fig. 6. The list of users $OL(\zeta)$ for the cells with at least one entry are shown in the table of Fig. 7. Here, the cell c_{10} is completely obstructed from the users u_1, u_2, u_3, where u_3 is stored in the user list of c_9, user u_2 is stored in the user list of its parent cell c_5, and so on.

Separate from the Quadtree, the information of the obstructed region (the collection of polygons) for each user is also stored. Each leaf level cell of the Quadtree is associated with a pointer to the corresponding obstructed regions that intersect with the cell. These obstructed regions are later used to compute the exact visibility of the necessary objects and candidates.

Auxiliary Quadtree partitioning: The purpose of maintaining this structure is to efficiently find the visible segments of an object w.r.t. a user. As the visibility of an object o is calculated by taking the summation of the visibility of its small segments $\Delta o.l$ of length at most ϵ, this value is used to direct the partitioning process. Specifically, if a cell ζ intersects with or contains any object $o \in O$, then ζ is further partitioned until the diagonal length of the cell ζ is less than or equal to the threshold ϵ.

8.2.2 OIR-tree

The set of objects O needs to be indexed in a way that the visibility of an object w.r.t. any user can be estimated in an efficient way. We extend the MIR-tree (Sect. 4) to support the visibility and textual similarity computation of our problem, as close-by objects (objects in an MBR) are likely to have a similar visibility value from a user, and it is efficient to estimate the distance, and angle (discussed in detail later) using the MBRs. We refer to this index as an OIR-tree. The OIR-tree is constructed in a similar manner to the original MIR-tree, where the MBRs of the line segments of the objects $o.l$ in O are used to construct the underlying R-tree. In addition, the following information is maintained:

– We maintain a reference to a cell of the auxiliary Quadtree with each node E of the OIR-tree, specifically, the cell ζ at the lowest level such that E is completely inside ζ. The idea is that if a node E is completely inside a cell ζ of the Quadtree, that means all of the objects stored in E are completely obstructed from the users in $OL(\zeta)$ (and the users stored in the OL of the ancestors of ζ), so E cannot contain any top-k object of those users.

Example 4 In Fig. 6, let the minimum bounding rectangle of the object o_3 be a node of the OIR-tree. This node intersects with the cells c_7 and c_8, but the cell c_2 is the lowest level cell for which this node is completely inside. Therefore, the reference to cell c_2 is associated with this node.

– The maximum and the minimum of the lengths of the objects stored in the subtree rooted at node E, denoted as $len^{\uparrow}(E)$ and $len^{\downarrow}(E)$, respectively, are stored.

Angle lookup table: A lookup table is maintained with the angle of each object $o \in O$ w.r.t. the Cartesian X-axis, denoted as $o.\theta$. This angle is later used to derive the angle of an object w.r.t. any user in query time.

8.3 Visibility bounds

Here we present an upper and a lower bound of visibility estimation of an object w.r.t. a user and the super-user using the node of the OIR-tree. Similar to the previous instance of MaxST with Euclidean distance as spatial similarity, these bounds are used to limit the number of top-k object computations of the necessary users and facilitate the pruning of the candidates. We also present bounds for the candidates to apply in the algorithms. The maximum (minimum) textual similarity $\mathrm{TS}^{\uparrow}(E.d, u.d)$ ($\mathrm{TS}^{\downarrow}(E.d, u.d)$) is computed from the maximum (minimum) TF-IDF value of the terms in $E.d \cap u.d$ in the same way as described in the previous sections.

8.3.1 Visibility bounds of an OIR-tree node of objects

Here, we present how to compute the maximum (minimum) visibility value of any object in a node E of the OIR-tree w.r.t. a user u such that $\forall o \in O, \overline{\mathrm{SS}}^{\uparrow}(E.l, u.l) \geq \overline{\mathrm{SS}}(o, u)$ ($\overline{\mathrm{SS}}^{\downarrow}(E.l, u.l) \leq \overline{\mathrm{SS}}(o, u)$).

Upper-bound visibility for a user: If a node E is completely inside the obstructed region of a user u, all the objects in E are also completely obstructed from u. Such a node E cannot contain any top-k object of u. As the visibility of an object depends on its visible length, distance, and angle w.r.t. a user, a straightforward approach to compute the upper-bound visibility of a node E for u is to use the maximum length of any object stored in E, $\mathrm{VL}^{\uparrow}(E.l, u.l) = len^{\uparrow}(E)$ as the visible length. The minimum Euclidean distance between u and E is used, and the angle is 90° (the angle for which the perceived length of any object is maximum). Although this bound is easy to compute, it is a loose bound, as the actual maximum *visible length* of any object in E can be very different from $len^{\uparrow}(E)$ as a result of object obstruction. Thereby, our techniques to compute a upper bound that estimates a tighter visible length are now presented.

From the reference of the Quadtree cell associated with E, first the leaf level cells of the Quadtree that intersect with E are found. Recall that a cell ζ is completely obstructed from the users in $OL(\zeta)$ and the users in the lists OL of the ancestors of ζ. Let $\chi_{E,u}$ be the set of the leaf level cells of the Quadtree that are both visible from u and intersects with at least one object in E. The angle lookup table that keeps the angle $o.\theta$ of each object o with the Cartesian x-axis is used to compute tighter upper-bound estimations as follows:

– First $o.\theta$ is used to better estimate the maximum length of a segment $\Delta o.l$ that is contained in a cell $x_i \in \chi_{E,u}$. The process is demonstrated in Fig. 8a. Let the length of the larger side of cell x_i be τ. As the length of a line segment with a slope inside a rectangle is maximum when it goes through a corner of a larger side of x_i, the maximum length of $\Delta o.l$ can be calculated as

$$\Delta o.l = \tau / \cos o.\theta.$$

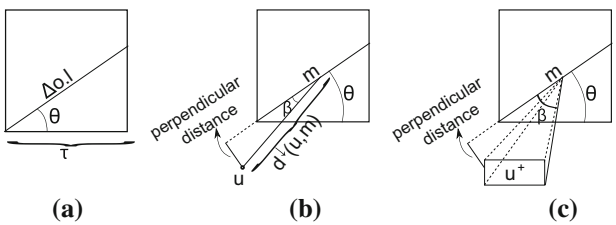


Fig. 8 Upper-bound calculation using angle. **a** Length, **b** angle, **c** angle w.r.t. u^{+}

– Now, a tighter upper bound of the angle between $o.l$ and u is estimated instead of using 90°. Recall Sect. 8.1, the small segments $\Delta o.l$ of size ϵ are partitioned such that the orientations of all points on $\Delta o.l$ w.r.t. to a user u can be considered as visually similar. Therefore, a point over $\Delta o.l$ is chosen, m. The midpoint of the line segment might be chosen, assuming that it goes through a corner of a larger side of x_i, but choosing any other point will have an insignificant visual difference. Then, the angle $\beta = \angle(\Delta o.l, u.l)$ can be calculated as

$$\sin\beta = \boldsymbol{d}_{\perp}(u.l, o.l)/\boldsymbol{d}^{\downarrow}(u.l, o.l).$$

The calculation is shown in Fig. 8b, where the perpendicular distance of $o.l$ from u is $\boldsymbol{d}_{\perp}(u.l, o.l)$.

– Finally, the minimum Euclidean distance between cell x_i and $u.l$ is used instead of the distance between E and u, the value of β as the upper bound of $\angle(\Delta o.l, u.l)$, and the value of $\Delta o.l$ is computed to get $\text{VL}^{\uparrow}_{\Delta}(\Delta o.l, u.l)$ for an individual segment. The $\text{VL}^{\uparrow}(E, u)$ is obtained by taking the summation of these values from the cells $\chi_{E,u}$. If this summation value is greater than the maximum length of an object in E, $len^{\uparrow}(E)$, then $len^{\uparrow}(E)$ is taken as the upper-bound visible length.

Upper-bound visibility for the super-user: As the algorithms to answer MaxST rely on the bounds of a node of objects for u^+ to filter the objects and candidates efficiently, the steps to compute the upper-bound visibility of a node E w.r.t. u^+, $\overline{\text{SS}}^{\uparrow}(E.l, u^+.l)$ such that $\forall u \in u^+$, $\forall o \in E$, $\overline{\text{SS}}^{\uparrow}(E.l, u^+.l) \geq \overline{\text{SS}}(o.l, u.l)$ are now presented.

Let χ_{E,u^+} be the set of the leaf level cells of the Quadtree that intersect with at least one object $o \in E$ and visible from at least one user $u \in u^+$. The upper bound w.r.t. u^+ is computed in a similar manner using χ_{E,u^+}. To calculate the upper bound of the angle w.r.t. the super-user, the angle is computed using the same technique mentioned above for each of the four corner points of the MBR of u^+. Let the maximum of these angles be $\beta = \angle^{\uparrow}(o.l, u^+)$, where β is between 0° and 90°. The angle of a line segment $\Delta o.l$ from any user location u inside the MBR of u^+ will be less than $\angle^{\uparrow}(o.l, u^+)$. This upper-bound calculation of angle is illustrated in Fig. 8c. The maximum visible length, the angle upper bound, and the minimum Euclidean distance are used to compute the final upper bound w.r.t. the super-user.

Lower-bound visibility: Let $\hat{\chi}_{E,u}$ be the set of the leaf level cells that are *obstructed* from u and intersects with at least one object $o \in E$. The maximum length of a segment $\Delta o.l$ that is contained inside a cell $\hat{x_i} \in \hat{\chi}_{E,u}$ is computed in the same manner as the upper- bound calculation. As these segments are obstructed, the sum of the maximum lengths of $\Delta o.l$ from $\hat{\chi}_{E,u}$ represent the maximum length of $o.l$ obstructed from u. Therefore, if this maximum obstructed length of $o.l$ is subtracted from $len^{\downarrow}(E)$ (the minimum length of any object in E), a lower bound of the visible length of $o.l$ is obtained. If this value becomes negative for any $o \in E$, the lower-bound visibility of E, $\overline{\text{SS}}^{\downarrow}(E.l, u.l)$, is taken as 0. Otherwise, to get a lower bound on the angle, first the angle $\beta = \angle(\Delta o.l, u.l)$ is computed for the object o that intersects with a cell $x_i \in \chi_{E,u}$ as above. If multiple objects intersect with x_i, then the minimum of their angles, $\angle^{\downarrow}(\Delta o.l, u.l)$, is used. The lower bound of the visible length, the angle $\angle^{\downarrow}(\Delta o.l, u.l)$, and the maximum Euclidean distances between E and u are used to compute lower-bound visibility with Eqs. 5 and 4.

Lower-bound visibility for the super-user: This bound is calculated in the same way as for an individual user, except that the calculation is done using the set $\hat{\chi}_{E,u^+}$ of the leaf level cells that are *obstructed* from at least one user $u \in U$, and intersecting with at least one object $o \in E$.

8.3.2 Visibility bounds of the candidates

Similar to the unobstructed instance of the MaxST problem, the upper (lower) bound of a candidate is computed by assuming the maximum (minimum) text similarity that can be achieved from the candidate keyword set and the keyword constraints. Now we present the techniques to calculate the upper- and the lower-bound visibility of a candidate location $\ell \in L$ w.r.t. a user u and the super-user.

Upper-bound visibility: Let the set of the leaf level cells that intersect with ℓ and visible from u be $\chi_{\ell,u}$. Note that the diagonal length of these cells can be larger than ϵ. Unlike the upper-bound calculation for an object, the length of ℓ inside each cell $x_i \in \chi_{\ell,u}$ can be easily calculated using the techniques to find the intersection points of a line segment and a rectangle. For each x_i, the length of ℓ inside x_i, the minimum distance between x_i and u, and the actual angle between the portion of ℓ inside x_i and u are used to calculate an upper bound of the visible length of that portion of ℓ. Finally, the sum of the upper bounds of visible lengths is taken to get the upper bound of the visibility of ℓ w.r.t. u using Eq. 4.

To calculate the upper-bound visibility w.r.t. u^+, similar to the calculation for an object, the set χ_{ℓ,u^+} of the leaf level cells that intersect with ℓ and visible from at least one user $u \in U$ are obtained. Similarly, the upper bound of the angle is the maximum of the angles calculated from the four corner points of u^+. Then, the exact length of ℓ inside each cell $x_i \in \chi_{\ell,u^+}$ is computed. These values and the minimum distance between ℓ and u^+ are used to get the upper-bound visibility of a candidate w.r.t. the super-user.

Lower-bound visibility: Let $\hat{\chi}_{\ell,u}$ be the set of the leaf level cells that are *obstructed* from u and intersect with the candidate ℓ. The length of ℓ that are contained inside each cell $\hat{x_i} \in \hat{\chi}_{\ell,u}$ is computed. These obstructed lengths are sub-

tracted from the actual length of ℓ to find the visible length of ℓ w.r.t. u. Then, the lower bound is calculated using the maximum Euclidean distance and the actual angle between ℓ and u. For the lower bound w.r.t. the super-user, the calculation is done similarly, but using the set of the leaf level cells that are *obstructed* from at least one user $u \in U$.
Computing the final visibility of a candidate: If a candidate line cannot be pruned using the bounds, the actual visibility w.r.t. some users may need to be computed. Recall from the construction of the auxiliary Quadtree that each leaf level cell is associated with a pointer to the corresponding obstructed regions (the collection of polygons) that intersect with the cell. If the diagonal length of a cell $x_i \in \chi_{\ell,u}$ is greater than ϵ, the obstructed regions of u are retrieved that intersect with x_i by following the pointer to find the segments of ℓ that are actually visible from u.

9 Experimental evaluation

In this section, the experimental evaluation for our three proposed algorithms is presented, (i) GRP- TOPK, (ii) INDIV- U, and (iii) INDEX- U approach to process the MaxST query for the two instances of spatial relevance in the ranking of an object: (i) the Euclidean distance and (ii) the visibility. We also compare our approaches with a straightforward baseline, where the top-k objects of each user is computed individually, and after some basic filtering, all the possible combination of the candidates are checked against the users to find the best candidate combination (Sect. 3).
Datasets and query generation: All experiments were conducted on three real datasets: (i) the Flickr dataset,[1] (ii) the LA dataset,[2] and (iii) the Yelp dataset.[3]

The LA dataset contains the 2D footprint of 542, 310 buildings in Los Angeles. The text description of 31, 526 POIs is collected from Foursquare for this area, and each text description is assigned to the corresponding building of that POI.

For the Flickr dataset, a total of 1 million images that are geotagged and contain at least one user-specified tag were extracted from the collection. The locations and tags are used as the location and keywords of the objects. For the experiments on visibility, the point locations are converted to rectangles (representing building footprints), where the distributions of the size of the rectangles follow the same distribution of the LA dataset.

The Yelp dataset contains information about businesses in 10 cities. For each business, three types of information are available: business location, business attributes, and user reviews on businesses. The attributes and reviews for each business are combined as the text description of that business. Similar to the Flickr dataset, the point locations are converted to building footprints for the visibility experiments. Table 4 lists the properties of the datasets.

We used the above datasets to generate the set of queries as follows. First, an area of a percentage of the dataspace size (here, 4%) was chosen, and $|U|$ number of objects in that area are taken randomly. Let this set of objects be O_u. The locations of the objects were used as the locations of the users. Then, keywords with the highest TF-IDF score were selected from O_u as the set of the user keywords, where UW was the number of unique user keywords. These keywords were distributed among the users such that each user had $|UL|$ number of keywords following the same distribution of keywords of O_u. In this work, we generated 50 such sets of users and reported the average performance.

Setup: All indexes and algorithms were implemented in Java. The experiments were ran on a 24 core Intel Xeon $E5-2630$ running at 2.3 GHz using 256 GB of RAM, and 1TB 6G SAS 7.2K rpm SFF (2.5-inch) SC Midline disk drives running Red Hat Enterprise Linux Server release 7.2 (Maipo). The Java Virtual Machine Heap size was set to 4 GB. All index structures are disk resident. The number of postings in the inverted lists was set to 128, and the page size was fixed at 1 kB for both indexes.

As multiple layers of cache existed between a Java application and the physical disk, we report simulated I/O costs in the experiments instead of physical disk I/O costs. The number of simulated I/O operations is increased by 1 when a node of a tree is visited. When an inverted list is loaded, the number of simulated I/O operations is increased by the number of blocks contained in the list. In the experiments, the performance was evaluated using cold-start queries.

9.1 Performance evaluation

In this section, we evaluate and compare the performance of the approaches by varying several parameters. The performance evaluation of our proposed approaches consists of the following two components:

- Computing the top-k objects for the users, where (i) the baseline (BL) approach computes the top-k objects for all of the users individually; (ii) the GRP- TOPK approach groups the users and computes the top-k objects of all the users jointly (GRP); and (iii) the INDIV- U approach that uses the same joint processing techniques of the GRP- TOPK approach, but uses the candidates to avoid computing the top-k object for some users (Indiv).
- Finding the best combination of the location and the keywords from the given set of candidates, where we

[1] http://webscope.sandbox.yahoo.com/catalog.php?datatype=i&did=67

[2] http://egis3.lacounty.gov/dataportal/2011/04/28/countywide-building-outlines

[3] http://www.yelp.com.au/dataset_challenge

Table 4 Description of dataset

Property	Flickr	LA	Yelp
Total objects	1,000,000	542,310	61,185
Total unique terms	166,317	52,731	266,869
Avg unique terms per object	6.9	8.5[a]	398.7
Total terms in dataset	6,936,385	274,577	77,838,026

[a]The avg unique terms of the objects (buildings) that are associated with a text description

Table 5 Parameters

Parameter	Range
k	5, **10**, 20, 50, 100
α	0.1, 0.3, **0.5**, 0.7, 0.9
No. of keywords per user, UL	1, 2, **3**, 4, 5, 6
No. of total unique keywords of users, UW	5, 10, **20**, 30, 40
Users' MBR as latitude × longitude, $Area$	1, 2, **4**, 8, 16
No. of candidate locations, $\|L\|$	1, 20, 50, **100**, 300
ω	1, 2, 3, 4, **5**, 6, 7, 8
No. of users, $\|U\|$	**100**, 500, $1K$, $2K$, $4K$

compare the performances of the exact (E) and the approximate (A) methods.

As the top-k object computation and the candidate selection steps are interleaved in the INDEX- U approach, the performance of the INDEX- U is reported as the total cost when varying the number of users at the end of this section.

The parameter ranges are listed in Table 5 where the values in bold represent the default values. In all experiments, a single parameter is varied while keeping the rest as the default settings to study the impact on: (i) the mean runtime per user (MRPU) to compute the top-k objects; (ii) the mean I/O cost per user (MIOPU) to compute the top-k objects; (iii) the total runtime of selecting the best candidate; and (iv) the approximation ratio of the approximate and the exact methods. This value is the ratio between the number of RkNN users of the best candidate returned by the approximate method, over the number of RkNN users of the best candidate returned by the exact method. Scalability is evaluated by varying the total number of users and by reporting (i) the total runtime and (ii) the total I/O cost for computing the top-k objects, instead of the mean values. We also evaluate the approaches for both the Euclidean distance and the visibility (V) as the spatial relevance in the MaxST problem. The runtime of all experiments is reported in milliseconds (ms).

Varying k: Figures 9 and 10 show the mean costs of computing the top-k objects of the users for Flickr and LA datasets, respectively. Since the GRP- TOPK and the INDIV- U approaches use several pruning strategies and avoid visiting any page multiple times, the costs are significantly lower than the baseline (BL). Both the mean I/O costs and the mean runtime per user are slightly less for the INDIV- U approach than the GRP- TOPK approach, as the INDIV- U approach prunes some users as well, where the GRP- TOPK approach computes the top-k objects for all users jointly.

The costs in the LA dataset are much higher than the costs in the Flickr dataset for the same settings, as half a million data points are densely located in LA, whereas the Flickr dataset contains 1 million data points scattered all over the world. Therefore, many objects in the LA dataset have very close similarity values for a user compared to the other dataset and must be retrieved in the process.

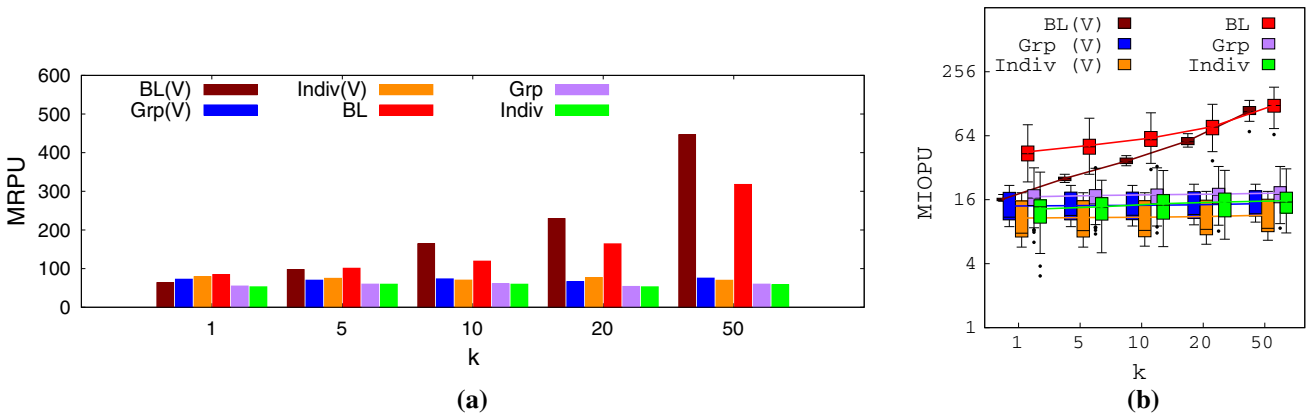


Fig. 9 Effect of varying k for Flickr dataset. **a** Runtime for finding top-k, **b** I/O cost for finding top-k

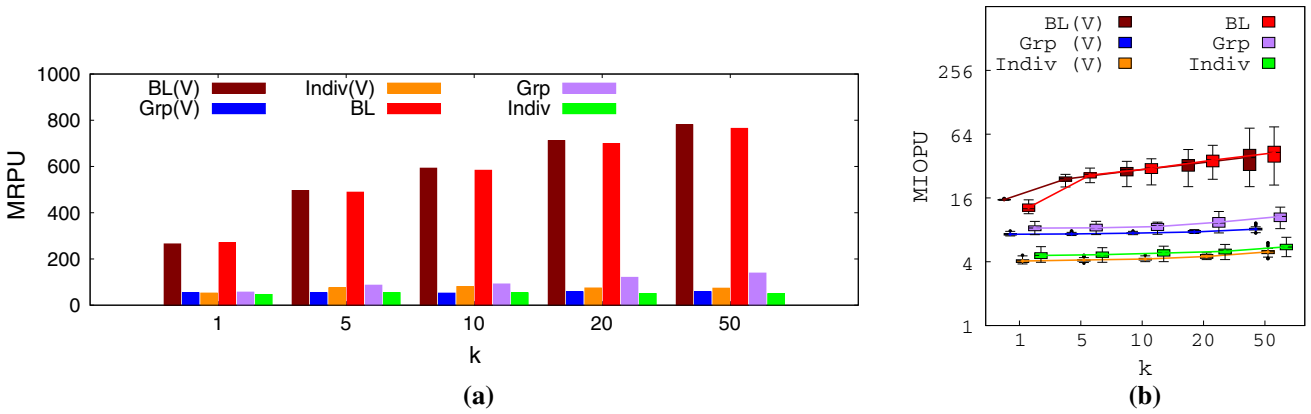


Fig. 10 Effect of varying k for LA dataset. **a** Runtime for finding top-k, **b** I/O cost for finding top-k

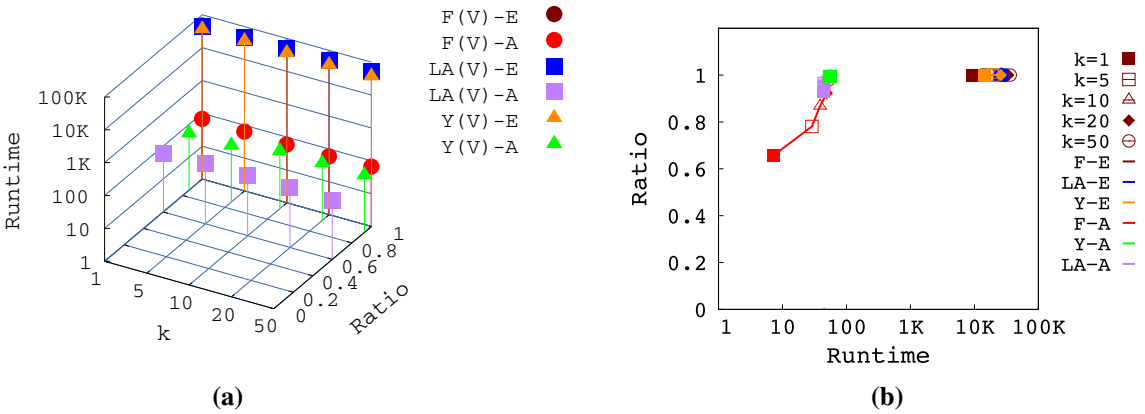


Fig. 11 Candidate selection runtime and approximation ratio trade-off for varying k. **a** Candidate selection runtime and approx. ratio (for visibility), **b** runtime versus approx. ratio (for Euclidean distance)

Although the trend of the changes in the costs is similar for the Euclidean distance and the visibility, the mean runtime per user for visibility is much higher than the Euclidean distance metric, as there are additional calculations required (finding the visible segments of a node/object and calculating angle bounds) for visibility.

Figure 11a and b demonstrates the trade-off between the runtime and the quality of the approximation to select the best candidate combinations for visibility and the Euclidean distance, respectively, in all three datasets. For both cases, the runtime of the exact (E) method does not vary much for k as it uses only basic pruning techniques and exhaustively computes all candidate combinations. The approximation ratio of the exact method is shown as 1 (the best ratio) to help compare with the approximate method. The runtime and the accuracy of the approximation increase with k, as more candidates are eligible to be included in the answer of the MaxST query. For all three datasets, the approximate method (A) selects the candidate combination of keywords greedily and thus requires around 3 orders of magnitude less computational time than the exact method.

Varying α: A higher value of α indicates more preference toward spatial similarity. As shown in our previous work [8], as the MBR of the users' locations and the union of the users' keywords remain the same, the cost of the top-k computation of our proposed approaches remains almost constant when α is varied. Figure 12 shows the runtime and the approximation quality when varying α. The runtime of each method does not vary much w.r.t. α.

The approximation ratio in the LA dataset is comparatively low for lower values of α, but rapidly increases as α increases. The reason is that as the location density of the objects in LA dataset is very high, the visibility scores (spatial similarity) for most of the relevant objects are usually very low. Therefore, the total similarity score of an object is more sensitive to the textual score (thereby, the choice of the candidate text) in the LA dataset when compared to the other datasets. Therefore, the accuracy of the approximate method increases rapidly as α increases for the LA dataset when a higher weight is given to the spatial similarity.

Varying *UL*: We now vary the number of keywords per user and present the effect on performance in Figs. 13 and 14

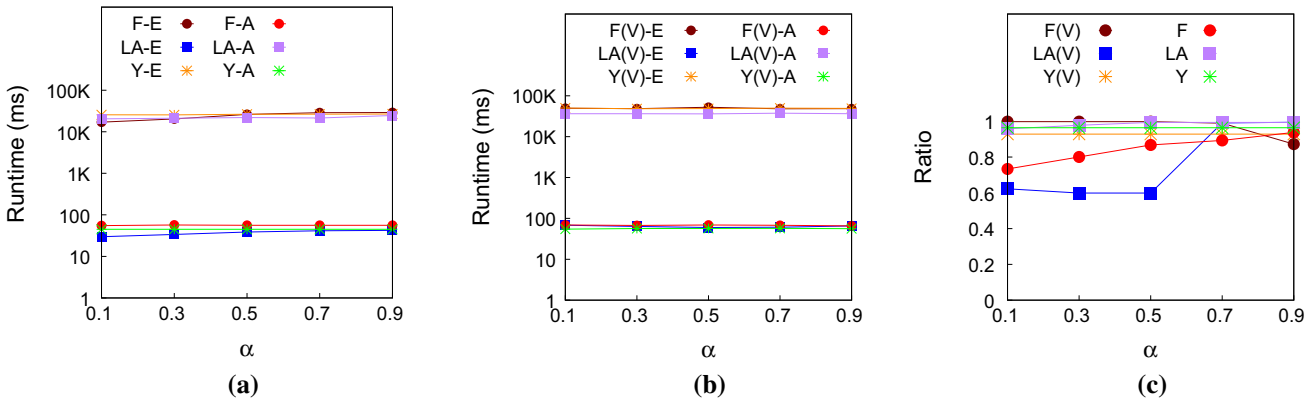


Fig. 12 Effect of varying α on candidate selection. **a** Runtime for Euclidean distance, **b** runtime for visibility, **c** approximation ratio

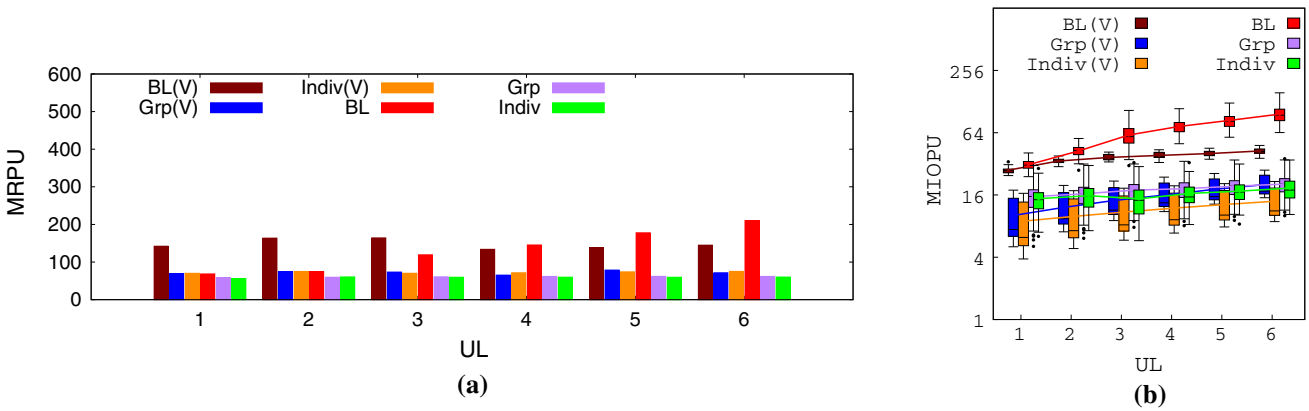


Fig. 13 Effect of varying UL for Flickr dataset. **a** for finding top-k, **b** I/O cost for finding top-k

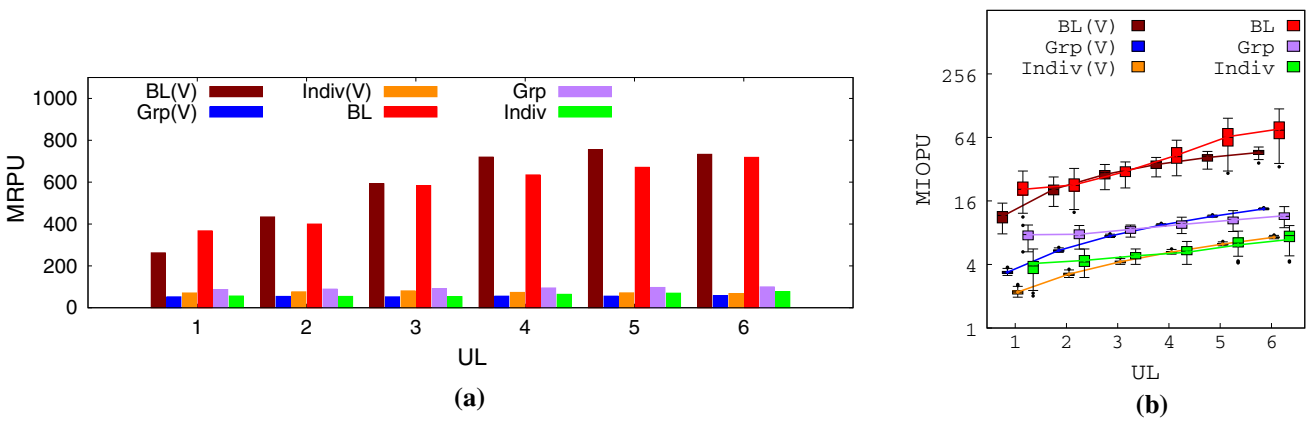


Fig. 14 Effect of varying UL for LA dataset. **a** Runtime for finding top-k, **b** I/O cost for finding top-k

for the Flickr and LA datasets, respectively. The cost of the baseline increases proportionally with the increase in UL for the Euclidean distance metric, as more objects become relevant to each user. The increase in the costs of the baseline is not that prominent for the visibility, as those additional objects may not be visible to that user. The mean costs of our proposed GRP- TOPK and INDIV- U approaches, where the users are grouped together as a super-user, do not vary much. The reason is that although UL increases, the total number of unique keywords in the group (UW) remains constant, so the number of objects retrieved remains unchanged as well.

Figure 15 shows the runtime and the approximation ratio to find the result candidate combination. Here, the number of users that are a reverse kNN of the result increases with the increase of UL for both exact and approximate methods, and the approximation quality increases as well.

Varying *UW*: Figure 16 shows the effect on performance when varying the total number of unique keywords for the group of the users in the Flickr dataset. Here, a lower value indicates that the queries share more keywords. As the GRP-TOPK and the INDIV- U approaches exploit shared I/Os among users, it outperforms the baseline, and the benefit is greater as overlap increases.

Figure 17a and b shows the runtime of the exact and the approximate approaches for selecting the candidate in all three datasets for the Euclidean distance and the visibility as spatial relevance, respectively. As the set of keywords UW is also the set of candidate keywords, the runtime of candidate selection increases for both methods. As the exact method checks all possible combinations of keywords after applying some basic pruning, the runtime for the exact method increases rapidly when compared to the approximate method. The runtime for visibility metric is more than that of the Euclidean distance for the same settings, as each visibility calculation is more computationally costly than a single Euclidean distance calculation.

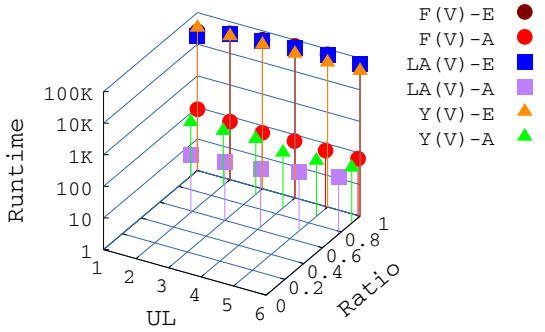


Fig. 15 Selection runtime and approximation ratio trade-off for varying *UL*

Figure 17a shows the effect on the approximation ratio when varying UW for all three datasets and for both spatial relevances. As UW increases, the number of combinations of the candidate keywords also increases. Therefore, the accuracy of the approximate method is very high for lower values of UW (almost 1) and decreases gradually as UW increases.

Varying ω: The performance when varying the number of candidate keywords to select ω is shown in Fig. 18. As the number of keyword combinations increases with ω, the runtime of both the exact and the approximate approach increases for both spatial similarity metrics (Fig. 18a, b). As the number of combinations to check in the exact method

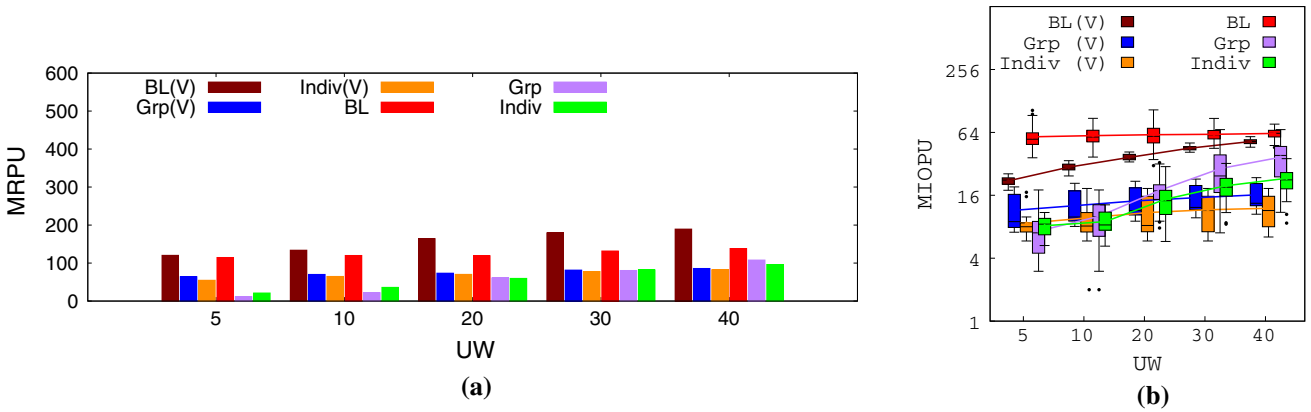


Fig. 16 Effect of varying *UW* for Flickr dataset. **a** Runtime for finding top-*k*, **b** I/O cost for finding top-*k*

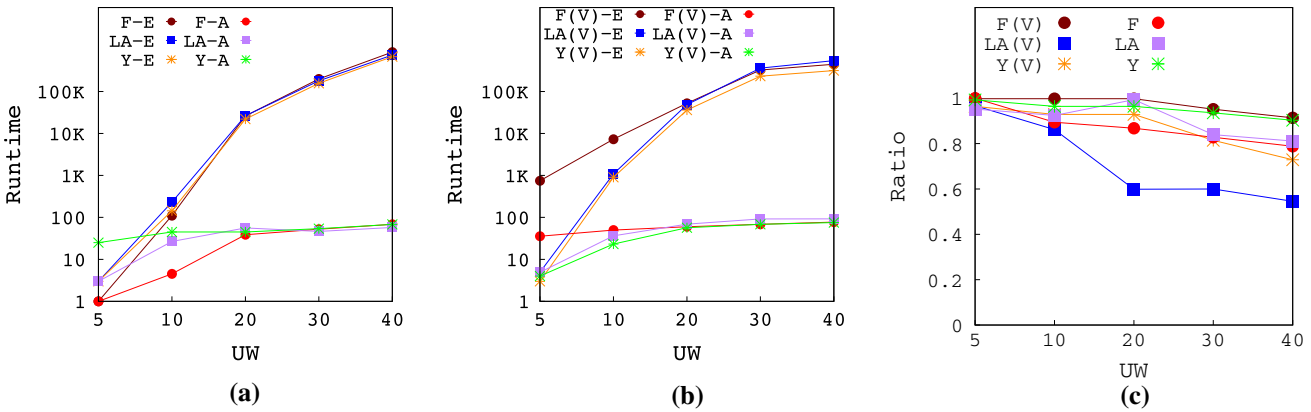


Fig. 17 Effect of varying UW on candidate selection. **a** for Euclidean distance, **b** runtime for visibility, **c** approximation ratio

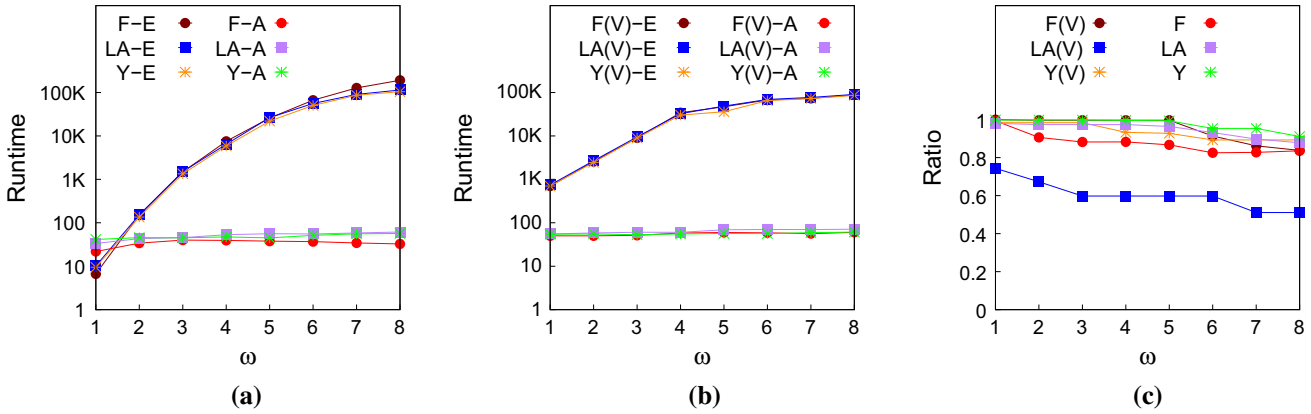


Fig. 18 Effect of varying ω for candidate selection. **a** Runtime for Euclidean distance, **b** runtime for visibility, **c** approximation ratio

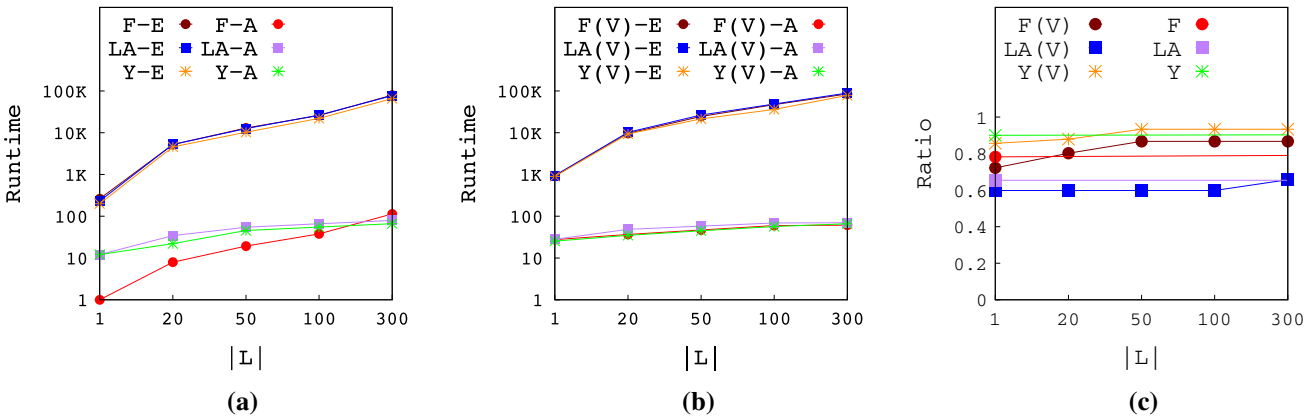


Fig. 19 Effect of varying $|L|$ for candidate selection. **a** Runtime for Euclidean distance, **b** runtime for visibility, **c** approximation ratio

increases exponentially with the increase of ω, the benefit of the approximate approach is higher as ω increases. The number of users that are a reverse kNN of the result candidate combination increases as ω increases, and the accuracy of the approximate method gradually drops. Figure 18c demonstrates this scenario for all of the datasets.

Varying $|L|$: Figure 19a and b shows the runtime when varying the number of candidate locations $|L|$ for the two different spatial similarity metrics, respectively. As the top-k processing of the users does not depend on $|L|$, we have only shown the performance of the exact and the approximate methods. As the total number of possible candidate combinations increases with the increase of $|L|$, the runtime increases proportionally with $|L|$ for both methods. As the visibility needs to be calculated for each qualifying candidate location, the runtime for visibility is higher than that for the Euclidean distance. The accuracy of the approximation increases slightly for a higher value of $|L|$, as more candidate locations become potential results.

Varying *Area*: The performance of the approaches does not vary much with the change in the area of the users' locations, which is similar to our prior finding [8] for Euclidean distance. The graphs are omitted for brevity.

Varying $|U|$: We evaluate the scalability of our proposed approaches by varying the number of users $|U|$. We show the runtime and the I/O costs of computing the top-k objects instead of the mean values for the LA dataset in Fig. 20. Similar to our previous finding in [8], as the number of users increases, the cost of the baseline increases proportionally as BL processes the users one by one. As the I/Os are shared among users, the advantage of our proposed approaches becomes more prominent when the number of users is higher.

Figure 21 shows the performance of the INDIV- U and the INDEX- U approaches as the total runtime of answering a MaxST query, including both the top-k object computation and the candidate selection time of the approximate approach. Both approaches apply pruning techniques to avoid computing the top-k objects of the unnecessary users, where the INDIV- U approach applies the techniques over individual users, and the INDEX- U approach finds them using a MIUR-tree index of the users. In all cases, the INDEX- U approach outperforms INDIV- U, as a hierarchy of the users helps to prune branches of the index (multiple users together), thus

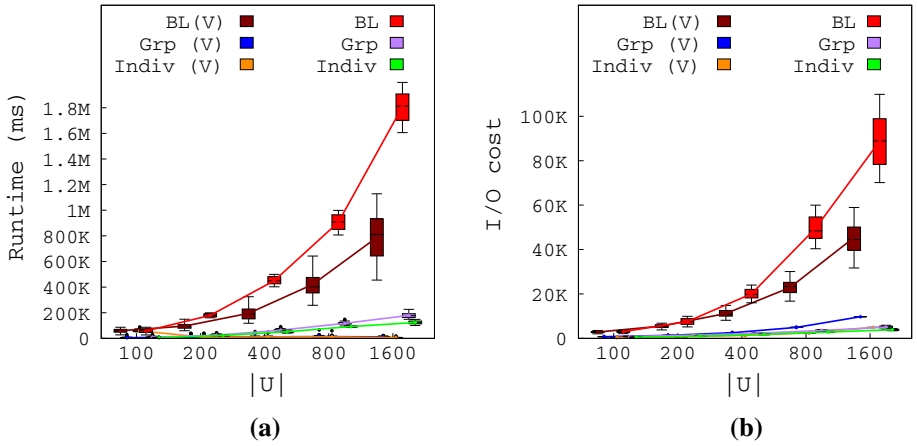


Fig. 20 Effect of varying $|U|$ for LA dataset. **a** for finding top-k **b** I/O cost for finding top-k

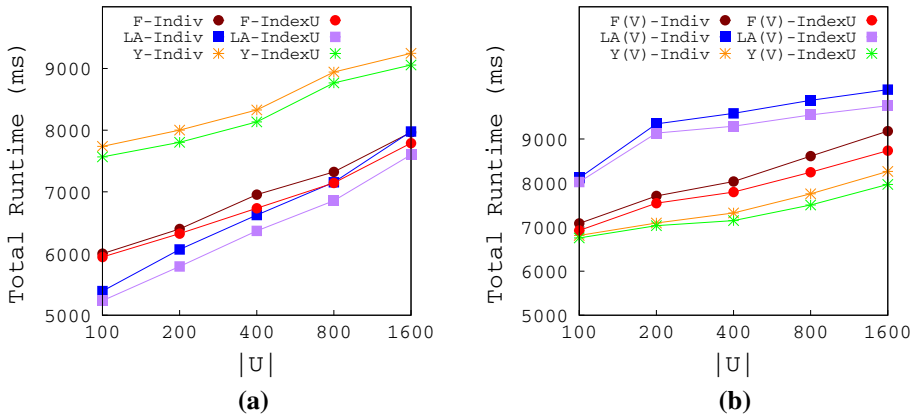


Fig. 21 Effect of user pruning by varying $|U|$. **a** Runtime for Euclidean distance, **b** runtime for visibility

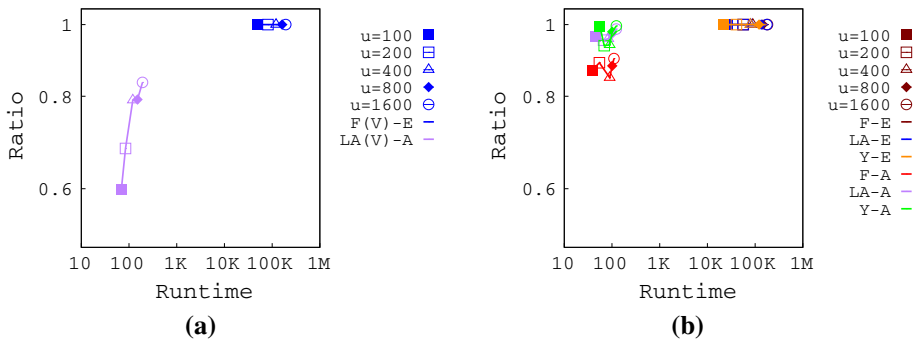


Fig. 22 Candidate selection runtime and approximation ratio trade-off for varying $|U|$. **a** Runtime versus approx. ratio (for visibility), **b** runtime versus approx. ratio (for Euclidean distance)

reducing overall time. The benefit of the INDEX- U approach slightly improves with the increasing value of $|U|$.

Efficiency versus approximation quality trade-off: Figure 22a and b shows the efficiency versus the approximation quality trade-off for varying the number of users for visibility and Euclidean distance, respectively. Figure 22a presents the trade-off in the LA dataset, and Table 6 presents the performance of the approaches for visibility in all of the three datasets. Clearly, the approximate approach is about 3–4 orders of magnitude faster than the exact method, as the approximate approach selects the candidate combination of keywords greedily.

In Fig. 22a, the approximation ratio in the LA dataset is comparatively low for lower values of $|U|$, but the approximation ratio does not vary much for the other two datasets (Table 6). This can be explained similarly when the value of α is varied using Fig. 12c. As the LA dataset is highly dense, the visibility score for most of the relevant objects is usually very low, and thus, the total similarity score of an object is more sensitive to the textual score for the default value of α (0.5) than the other datasets. As $|U|$ increases, the greedy selection of the candidate keywords can increase relevance with more users. The advantage of approximation in both efficiency and quality is more prominent as $|U|$ increases.

Table 6 Performance of approximate approach for varying $|U|$ (visibility as spatial relevance)

Dataset		$\|U\|$				
		100	200	400	800	1600
Flickr	Runtime (E)	52K	82K	121K	180K	190K
	Runtime (A)	59.9	73.4	99.9	121.3	145.3
	Ratio (A)	0.99	0.98	0.98	0.95	0.97
	TR (E)	0.86	0.78	0.68	0.53	0.50
	TR (A)	1.00	0.99	0.99	0.97	0.99
LA	Runtime (E)	48K	81K	120K	161K	192K
	Runtime (A)	69.0	84.4	120.2	150.4	191.9
	Ratio (A)	0.60	0.68	0.79	0.79	0.84
	TR (E)	0.87	0.79	0.69	0.58	0.50
	TR (A)	0.80	0.84	0.90	0.90	0.92
Yelp	Runtime (E)	36K	67K	100K	164K	189K
	Runtime (A)	56.7	69.5	85.5	91.3	128.4
	Ratio (A)	0.93	0.94	0.97	0.97	0.96
	TR (E)	0.90	0.82	0.73	0.57	0.50
	TR (A)	0.96	0.97	0.98	0.98	0.98

To better comprehend the trade-off between efficiency and the approximation quality, we present a trade-off score TR, where the runtime and the approximation ratio are combined by a linear weighted function. Specifically,

$$\text{TR} = \gamma \times T + \gamma \times \text{Ratio},$$

where T is the runtime normalized between [0, 1]. We show the trade-off scores for both exact and approximate methods in Table 6 for $\gamma = 0.5$, i.e., equal weight on both values.

10 Related work

10.1 Spatial databases

Relevant work from the spatial database domain can be categorized mainly as *Maximizing Bichromatic Reverse k Nearest Neighbor* (MaxBRkNN) and location selection queries. MaxBRkNN queries: Wong et al. [33] introduced the MaxOverlap algorithm to solve the MaxBRkNN problem. The algorithm iteratively finds the intersection point of the Nearest Location Circles (NLCs) that are covered by the largest number of NLCs. The optimal region is the overlap of these NLCs. This work also supports ℓ-MaxBRkNN queries to find the ℓ best regions. In a later work, [34] extended the MaxOverlap algorithm to support the L_p-norm and three-dimensional space. However, the scalability of MaxOverlap is an issue, as the computation of the intersection points for the NLCs is expensive.

Other work exists that overcome the limitations of MaxOverlap. Zhou et al. [43] introduced the MaxFirst algorithm which iteratively partitions the space into quadrants and used the NLCs to prune the quadrants that cannot be a part of the result. Liu et al. [20] present the MaxSegment algorithm that transforms the optimal region search problem to the optimal interval search problem. They use a variant of plane sweep to find the optimal interval.

Approximate solutions have also been proposed to improve the efficiency. Lin et al. [18] proposed OptRegion where each NLC is approximated by the minimum bounding rectangle (MBR) and a sweep-line technique is used to find the overlapping NLCs. An estimation of the number of overlapping NLCs is computed using the MBRs to prune the intersection points. Alternately, Yan et al. [38] propose a grid-based approximation algorithm called FILM. Since the algorithm is approximate, the solution requires an order of magnitude less computation time than MaxOverlap. The authors extended FILM to answer the related problem of locating k new services that collectively maximize the total number of users.

These previous studies focus solely on spatial properties such as the intersection of geometric shapes [33,34], space partitioning [43], or sweep-line techniques [18,20] in the query processing methods. Therefore, it is not straightforward to extend these solutions to support the textual component of the MaxST query.

Location selection queries: The work [27,41] explores optimal location queries, which find a location for a new facility that minimizes the average distance from each customer to the closest facility. Zhang et al. [41] propose the *MDOL prog* algorithm which partitions space to find an optimal location. Qi et al. [27] maintain the *influence set* of a potential location p that includes the customers for whom the nearest facility distance is reduced if a new facility is established at p. A similar problem was presented in other work [1,28,39,40] which finds a location for a new server such that the maximum distance between the server and any client is minimized. Papadias et al. [26] find a location that minimizes the sum of the distances from the users. The optimal location query is also explored for road network distance [2,5,36]. The proposed approaches exploit the assumption that the location of the objects is confined by the underlying road network and thus reduce the number of computations significantly than the approaches proposed for Euclidean distance.

A *maximal influence query* ([4,17,19,31,35]) finds the optimal location to place a new facility such that the influence of that facility is maximized. Here, the influence of a location c represents the cardinality of customers whose corresponding nearest facility will be c if a new facility is established in c. These queries focus on an aggregation over distances from the query location, such as the average or the minimum

distance. These works do not directly address the problem of this article, maximizing the reverse kNN users.

10.2 Spatial–textual databases

In the literature of spatial–textual queries, the reverse spatial–textual kNN (RSTkNN) and the maximizing reverse spatial–textual kNN (MaxST) are the most relevant to our problem.

RSTkNN: Given a dataset D of spatial–textual objects, a target query object q, an RSTkNN query finds all the objects in D that have q in their list of top-k relevant objects. The ranking of the objects uses an objective function which combines both spatial proximity and text relevancy. Lu et al. [21] proposed the Intersection-Union R-tree (IUR-tree) index and later presented a cost analysis for RSTkNN queries [22]. Each node of an IUR-tree consists of an MBR and two textual vectors: an intersection vector and a union vector. The weight of each term in the intersection (union) textual vector is the minimum (maximum) weight of the terms in the documents that are contained in the corresponding subtree. Each non-leaf node is also associated with the number of objects in the subtree rooted at that node.

In their proposed solution, an upper- and a lower-bound similarity are computed between each node of the IUR-tree and the kth most similar object. A branch-and-bound algorithm is then used to answer the RSTkNN query. In this work, the computation of the bounds and the algorithm are designed for the monochromatic case only since both the data objects and the query objects belong to the same type, and the nodes of the tree store only one type of object.

MaxRSTkNN: Given a set of users and a set of facilities, [14] address the problem of selecting at most ω keywords as the text description of a specific facility, such that the facility will appear in the top-k results of the maximum number of users. An extension of this problem is studied in our earlier work ([8]), where the problem is to select both the location and the text description to establish a facility so that the reverse spatial–textual kNN of that facility from the set of users will be maximum.

A recent work by [37] independently proposes a solution of a subproblem of this article, where, given a set of keywords, the query is to find a region in space to establish a facility such that the facility will be a top-k spatial–textual object of the maximum number of users. They present both an exact and an approximate solution based on Voronoi diagrams to find such regions.

10.3 Visibility queries

Visibility problems studied in spatial databases usually involve finding the kNN [25,32] or RkNN [11–13] objects for a given query point, where the shortest path between two points without crossing any obstacle is taken as the distance measure, denoted as the *obstructed distance*.

Masud et al. [24] propose the *k maximum visibility query* that finds top-k locations from a set of query locations with the maximum visibility of a target object T in the presence of obstacles. Choudhury et al. [7] present an efficient approach to construct a Visibility Color Map (VCM), where each point in the space is assigned a color value denoting the visibility measure of a query target. Rabban et al. [29] study the problem of constructing a VCM for a moving target. In general, all of these approaches use the concept of an obstructed region to find the parts of the space that are visible from the query location(s) and then calculate the visibility value of the visible part of the target object.

Visual–textual queries: Zhang et al. [42] study the problem of finding the top-k objects based on the visibility and the textual similarity with respect to a query location and a set of query keywords. They propose a two-pass method on the IR-tree where (i) the first pass iteratively explores the region around the query location to determine the obstructed and the visible region w.r.t. the query, and (ii) the second pass is used to calculate the visibility and textual similarity of the visible objects in a best-first manner.

Our approach to answer the MaxST query for visibility metric also follows a similar principle, where the obstructed regions are pre-computed and indexed, and then, an OIR-tree is used to compute the visibility of the necessary objects and the candidates.

11 Conclusion

In this work, we presented solutions to efficiently answer a *Maximized Bichromatic Reverse Spatial Textual k Nearest Neighbor* (MaxST) query, which finds an optimal location and a set of keywords for an object so that the object has the maximum number of RkNNs. We proposed three different solutions to answer the MaxST query for the spatial–textual data, and we presented the necessary calculations for the approaches using two different spatial similarity metrics: (i) the Euclidean distance and (ii) the visibility. In all of our proposed methods, we improved the overall efficiency by pruning the candidates, sharing the processing and I/O costs of the multiple users, and avoiding multiple retrievals of the same object, even for visibility, where the visibility calculation of an object requires the location of the other objects as well. We also proposed an approximate solution for the keyword selection component of the problem.

Through extensive experiments using three publicly available datasets, we compared the performance of our proposed approaches in different settings. In summary, we find that the approximate algorithm, which greedily selects the key-

words, is around 2–3 orders of magnitude faster than an exact method. The GRP- TOPK approach that groups the users and finds the top-*k* objects of the users jointly performs about 2–3 times better than the baseline, and the benefit increases when using a visibility metric. As the INDIV- U approach prunes users as well, the performance of the INDIV- U approach is close to or better than the GRP- TOPK approach in all settings tested. The INDEX- U performs slightly better than the INDIV- U approach, at the cost of maintaining an additional index for the users, but the benefits increase as the number of users increases.

Acknowledgements This work was supported by the Australian Research Council's *Discovery Projects* Scheme (Grants DP140101587, DP170102231, and DP170102726) and Google Faculty Research Award. This work was partially supported by the National Natural Science Foundation of China (NSFC) 91646204. Farhana Choudhury is the recipient of a scholarship from National ICT Australia. We thank Lisi Chen, Gao Cong, Christian S. Jensen, and Dingming Wu for providing the implementation of the IR-tree in [3].

References

1. Cardinal, J.J., Langerman, S.L.: Min-max-min geometric facility location problems. In: EWCG, pp. 149–152 (2006)
2. Chen, Z., Liu, Y., Chi-Wing Wong, R., Xiong, J., Mai, G., Long, C.: Efficient algorithms for optimal location queries in road networks. In: SIGMOD (2014)
3. Chen, L., Cong, G., Jensen, C.S., Wu, D.: Spatial keyword query processing: an experimental evaluation. PVLDB **6**(3), 217–228 (2013)
4. Chen, J., Huang, J., Wen, Z., He, Z., Taylor, K., Zhang, R.: Analysis and evaluation of the top-k most influential location selection query. Knowl. Inf. Syst. **43**(1), 181–217 (2015)
5. Chen, Z., Liu, Y., Wong, R.C.-W., Xiong, J., Mai, G., Long, C.: Optimal location queries in road networks. ACM Trans. Database Syst. **40**(3), 1–41 (2015)
6. Choudhury, F.M., Culpepper, J.S., Sellis, T.: Batch processing of top-k spatial-textual queries. In: GeoRich, pp. 7–12 (2015)
7. Choudhury, F.M., Ali, M.E., Masud, S., Nath, S., Rabban, I.E.: Scalable visibility color map construction in spatial databases. Inf. Syst. **42**, 89–106 (2014)
8. Choudhury, F.M., Culpepper, J.S., Sellis, T., Cao, X.: Maximizing bichromatic reverse spatial and textual k nearest neighbor queries. PVLDB **9**(6), 456–467 (2016)
9. Cong, G., Jensen, C.S., Wu, D.: Efficient retrieval of the top-k most relevant spatial web objects. PVLDB **2**(1), 337–348 (2009)
10. Feige, U.: A threshold of ln n for approximating set cover. J. ACM **45**(4), 634–652 (1998)
11. Gao, Y., Yang, J., Chen, G., Zheng, B., Chen, C.: On efficient obstructed reverse nearest neighbor query processing. In: GIS, pp. 191–200 (2011)
12. Gao, Y., Zheng, B., Chen, G., Lee, W.C., Lee, K.C.K., Li, Q.: Visible reverse k-nearest neighbor query processing in spatial databases. TKDE **21**(9), 1314–1327 (2009)
13. Gao, Y., Liu, Q., Miao, X., Yang, J.: Reverse k-nearest neighbor search in the presence of obstacles. Inf. Sci. **330**, 274–292 (2016)
14. Gkorgkas, O., Vlachou, A., Doulkeridis, C., Nørvåg, K.: Maximizing influence of spatio-textual objects based on keyword selection. In: SSTD, pp. 413–430 (2015)
15. Guttman, A.: R-trees: a dynamic index structure for spatial searching. In: SIGMOD, pp. 47–57 (1984)
16. Hochbaum, D.: Approximation Algorithms for NP-hard Problems. PWS Publishing Company, Boston (1997)
17. Huang, J., Wen, Z., Qi, J., Zhang, R., Chen, J., He, Z.: Top-k most influential locations selection. In: CIKM, pp. 2377–2380 (2011)
18. Lin, H., Chen, F., Gao, Y., Lu, D.: OptRegion: finding optimal region for bichromatic reverse nearest neighbors. In: DASFAA, pp. 146–160 (2013)
19. Lin, Q., Xiao, C., Cheema, M.A., Wang, W.: Finding the sites with best accessibilities to amenities. In: DASFAA, pp. 58–72 (2011)
20. Liu, Y., Wong, R.-W., Wang, K., Li, Z., Chen, C., Chen, Z.: A new approach for maximizing bichromatic reverse nearest neighbor search. Knowl. Inf. Syst. **36**(1), 23–58 (2013)
21. Lu, J., Lu, Y., Cong, G.: Reverse spatial and textual k nearest neighbor search. In: SIGMOD, pp. 349–360 (2011)
22. Lu, Y., Lu, J., Cong, G., Wu, W., Shahabi, C.: Efficient algorithms and cost models for reverse spatial-keyword k-nearest neighbor search. ACM Trans. Database Syst. **39**(2), 1–46 (2014)
23. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, Cambridge (2008)
24. Masud, S., Choudhury, F.M., Ali, M.E., Nutanong, S.: Maximum visibility queries in spatial databases. In: ICDE, pp. 637–648 (2013)
25. Nutanong, S., Tanin, E., Zhang, R.: Incremental evaluation of visible nearest neighbor queries. TKDE **22**(5), 665–681 (2010)
26. Papadias, D., Qiongmao, S., Yufei, T., Kyriakos, M.: Group nearest neighbor queries. In: ICDE, pp. 301–312 (2004)
27. Qi, J., Rui, Z., Kulik, L., Lin, D., Yuan, X.: The min-dist location selection query. In: ICDE, pp. 366–377 (2012)
28. Qi, J., Xu, Z., Xue, Y., Wen, Z.: A branch and bound method for min-dist location selection queries. In: ADC, pp. 51–60 (2012)
29. Rabban, I.E., Abdullah, K., Ali, M.E., Cheema, M.A.: Visibility color map for a fixed or moving target in spatial databases. In: SSTD, pp. 197–215 (2015)
30. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Inf. Process. Manage. **24**(5), 513–523 (1988)
31. Sun, Y., Huang, J., Chen, Y., Zhang, R., Du, X.: Location selection for utility maximization with capacity constraints. In: CIKM, pp. 2154–2158 (2012)
32. Wang, Y., Gao, Y., Chen, L., Chen, G., Li, Q.: All-visible-k-nearest-neighbor queries. In: DEXA, pp. 392–407 (2012)
33. Wong, R.C.-W., Özsu, M.T., Yu, P.S., Fu, A.W.-C., Liu, L.: Efficient method for maximizing bichromatic reverse nearest neighbor. PVLDB **2**(1), 1126–1137 (2009)
34. Wong, R.C.-W., Özsu, M.T., Fu, A.W.-C., Yu, P.S., Liu, L., Liu, Y.: Maximizing bichromatic reverse nearest neighbor for lp-norm in two and three-dimensional spaces. PVLDB **20**(6), 893–919 (2011)
35. Xia, T., Zhang, D., Kanoulas, E., Du, Y.: On computing top-t most influential spatial sites. In: VLDB, pp. 946–957 (2005)
36. Xiao, X., Yao, B., Li, F.: Optimal location queries in road network databases. In: *ICDE* (2011)
37. Xie, X., Lin, X., Xu, J., Jensen, C.: Reverse keyword-based location search. In: ICDE (2017). (**to appear**)
38. Yan, D., Wong, R.C.-W., Ng, W.: Efficient methods for finding influential locations with adaptive grids. In: CIKM, pp. 1475–1484 (2011)
39. Yan, D., Zhao, Z., Ng, W.: Efficient algorithms for finding optimal meeting point on road networks. PVLDB **4**(11), 968–979 (2011)
40. Yan, D., Zhao, Z., Ng, W.: Efficient processing of optimal meeting point queries in euclidean space and road networks. Knowl. Inf. Syst. **42**(2), 319–351 (2015)
41. Zhang, D., Du, Y., Xia, T., Tao, Y.: Progressive computation of the min-dist optimal-location query. In: VLDB, pp. 643–654 (2006)

42. Zhang, C., Shou, L., Chen, K., Chen, G.: See-to-retrieve: efficient processing of spatio-visual keyword queries. In: SIGIR, pp. 681–690 (2012)
43. Zhou, Z., Wu, W., Li, X., Lee, M.L., Hsu, W.: MaxFirst for MaxBRkNN. In: ICDE, pp. 828–839 (2011)
44. Zobel, J., Moffat, A., Ramamohanarao, K.: Inverted files versus signature files for text indexing. ACM Trans. Database Syst. **23**(4), 453–490 (1998)