Maximizing Influence of Spatio-Textual Objects Based on Keyword Selection

Orestis Gkorgkas[1], Akrivi Vlachou[1,2], Christos Doulkeridis[3], and Kjetil Nørvåg[1]

[1] Norwegian University of Science and Technology (NTNU), Norway
[2] Institute for the Management of Information Systems, R.C. "Athena", Greece
[3] Department of Digital Systems, University of Piraeus, Greece
{orestis,vlachou,noervaag}@idi.ntnu.no, cdoulk@unipi.gr

Abstract. In modern applications, spatial objects are often annotated with textual descriptions, and users are offered the opportunity to formulate *spatio-textual queries*. The result set of such a query consists of spatio-textual objects ranked according to their distance from a desired location and to their textual relevance to the query. In this context, a challenging problem is how to select a set of at most b keywords to enhance the description of the facilities of a spatial object, in order to make the object appear in the top-k results of as many users as possible. In this paper, we formulate this problem, called *Best-terms* and we show that it is NP-hard. Hence, we present a baseline algorithm that provides an approximate solution to the problem. Then, we introduce a novel algorithm for keyword selection that greatly improves the efficiency of query processing. By means of a thorough experimental evaluation, we demonstrate the performance gains attained by our approach.

1 Introduction

Spatio-textual search has attracted increased attention recently, due to the numerous applications that provide value-added services to the users by combining spatial location with textual relevance. Given a database of geographical points of interest that are annotated with textual information (also called *spatio-textual objects*), the objective of a spatio-textual query is to retrieve a ranked set of top-k spatio-textual objects that are close to the query point and have high textual similarity to the query keywords. As a notable example, consider hotels that are annotated with their facilities (e.g., in the form of keywords), and tourists that search for hotels close to some location of interest and a set of query keywords indicating desired facilities (for example "pool" or "Wi-Fi").

An interesting problem encountered in real-life applications that rely on spatio-textual retrieval is how to improve the ranking of a spatio-textual object for as many users as possible. For instance, for a newly established hotel at some location, the question is how to enrich its textual annotation in order to maximize its rank for many different users. To address this challenging problem, we capitalize on reverse top-k queries [19], which retrieve the set of users that

have a given object in their top-k results. We model the problem as a maximization of the cardinality of the reverse top-k result set, and we explore the different combinations of keywords that will increase the query object's rank for many users, when added to its textual annotation. We call this problem as *Best-terms*, we show that it is NP-hard, and we present a greedy solution that serves as baseline. Then, we propose a novel algorithm that boosts the performance of query processing, by deliberately selecting keywords that increase the score of the query object for many users simultaneously. Finally, we present the results of our experimental evaluation that verifies the performance gains of our algorithm.

In summary, our main contributions are outlined below:

- We formulate the novel problem, called Best-terms, of increasing the rank of a spatio-textual object for many different users, by enriching its textual description.
- We show that the Best-terms problem is NP-hard and we provide a baseline solution.
- We propose an efficient query processing algorithm that significantly outperforms the baseline consistently.
- We provide an experimental evaluation that demonstrates the merits of our approach.

The rest of this paper is structured as follows: Section 2 provides an overview of the related work. Section 3 presents the necessary background and preliminary concepts. Then, in Section 4, we formally describe the problem statement. Section 5 presents the baseline algorithm, while Section 6 describes our efficient query processing algorithm. Section 7 presents the experimental evaluation, and Section 8 concludes the paper.

2 Related Work

In this section, we provide an overview of the related research literature.

Keyword recommendation. Zhang et al. [23] present a method for recommending keywords for advertisements in keyword search results using Wikipedia. They focus mostly in cases where the advertisement (target) consists of short-text web pages that contain inadequate textual content to describe the advertised entity. Based on the fact that a large number of entities are described in Wikipedia, they use Wikipedia articles relevant to the advertised entity in order to recommend keywords to connect to the target. Fuxman et al. [9] follow a different approach. They suggest keyword queries to advertisers using logs that store the queries posed by the users and the URLs of the result set that were selected by the users. Some of the URLs are also connected to a set of concepts. The target of the authors is to connect the set of concepts to the queries using the Markov Random Field model and suggest the most relevant queries for each concept to the advertisers. Ravi et al. [16] propose a variety of methods for automatic generation of bid phrases. Among others they introduce the usage of a

translation model that extends a predefined mapping between bidding phrases and target web pages. Papadimitriou et al. [15] study the problem of mapping an advertisement in a set of URLs based on a set of keyword queries. In particular, they assume that each advertisement is mapped to a set of keyword queries and their aim is to map each advertisement in a set of URLs which will be representative of the results produced by the attached keyword queries. Choi et al. [4] create a representative summary of the advertisement based on the context of the advertised material. Their method makes use of co-occurrence and semantic vectors in order to enrich the ad context and create a representative set of terms. Cholette et al. [5] study the problem of finding optimal bids in search-based algorithms. Agrawal et al. [1] introduce an approach for recommending bid phrases from a given ad landing page by classifying a set of labels generated by click logs. Their classifier has logarithmic complexity and can efficiently make predictions on large sets of labels.

The aim of the aforementioned approaches is to identify potentially relevant queries to the advertised products and form bid phrases based on the identified queries. Our approach is inherently different, because the above techniques try to predict relevant queries and do not consider the relevance of the advertised product in relation to similar products. In addition, they do not consider top-k search criteria as the appearance of a product in a search result is decided mainly on the bidding strategy. On the contrary, our aim is to enhance the description of a spatio-textual object and to increase the number of queries for which the target product appears in the top-k list of the search results. In this effort, we take into consideration not only the user preferences, but also the rest of the spatio-textual objects that are relevant to those queries.

Spatial keyword search. Spatial keyword search has been well studied during the recent years and several index structures have been introduced for efficient search. A detailed evaluation of existing spatio-textual indexes can be found in [3]. Cong et al. [6] introduced the IR-tree and its variants. The IR-tree is based on the R-tree structure. Each node of the tree is also associated with inverted index containing the textual information of the children of the node. Rocha et al. [17] proposed the S2I index which uses different strategies for frequent and infrequent terms and outperforms the IR-tree. Zhang et al. proposed the I^3 index [22], which is based on the quadtree, and the RCA approach [21], which is based on Fagin's CA algorithm [8]. Both approaches outperform the S2I and IR-tree index structures. Nonetheless, the IR-tree is able to perform a spatial only search, retrieving objects that are not textually relevant to a spatio-textual query. This possibility is not offered by any of the S2I, I^3 and RCA approaches. Cao et al. [2] introduce the concept of *prestige* where a spatio-textual object has a higher prestige if it is collocated with other textually similar objects. They calculate the prestige of a spatio-textual object based on a graph where each node corresponds to an object and two nodes are connected if and only if their textual similarity and spatial proximity exceed certain thresholds. Deng et al. [7] suggested an approach of finding a set of spatio-textual objects that are relevant to a spatio-textual query and at the same time they fulfill a desired spatial

property. In particular, their aim is to identify a keyword-cover of optimal score, where as keyword cover is defined a set of objects where each object is associated with exactly one term of the spatio-textual query.

Lu et al. [14] and Lu et al. [13] studied the problem of reverse spatial and textual k nearest neighbor search where, given a query point q, the objective is to locate the set of spatio-textual objects for which q is among the k nearest neighbors. The distance between the objects is a linear combination of the textual and the Euclidean distance of the objects. The authors introduce the IUR-tree which is an adaptation of the IR-tree. Each node of the IUR-tree contains the union and the intersection of the terms contained in the objects in the subtree rooted at the node. Our approach is different, as we do not evaluate the similarity between elements of a set of spatio-textual objects, but our aim is to increase the relevance and therefore the visibility of an object against a set of user preferences which constitutes a different set from that of the spatio-textual objects that our query object belongs.

Wu et al. [20] propose the W-IR-tree which is similar to the IR-tree but it is constructed based primarily on textual distance. The W-IR-tree shows improved performance for batch queries where objects are considered relevant to the query only if they contain all terms of the query. The W-IR-tree cannot be applied in our case as we consider it possible for a spatio-textual object to be relevant to a user preference even if it does not contain all terms of the user preference. Gao et al. [10] propose a filter-and-refinement framework for processing reverse boolean top-k spatial keyword queries. They focus on queries where a spatio-textual object must contain all terms of a query to be considered a valid result. Lin et al. [12] study the problem of identifying important terms in the textual description of a spatio-textual object that cause the object to be highly ranked for a specific query or a specific region. Our approach is different, as we focus on enriching the textual description of a an object with new terms.

3 Preliminaries

Let D be a set of objects, where each object o is represented by a tuple of the form $o = \langle o.T, o.L \rangle$ where $o.T$ is a set of keywords describing the features of o and L is a point in $\mathbb{R}^2$ describing the location of o. We denote as $\mathcal{A} = \bigcup_{o \in D} o.T$ to be the set of all keywords in D. In the scope of this paper, we call these objects *spatio-textual objects*. For a given object o, we consider the *size* of o to be equal to $|o.T|$, namely the size of an object is the number of terms it contains.

3.1 Top-k spatial keyword queries

Let u be a user preference query on D, where u is represented by the a tuple $u = \langle u.T, u.L, \alpha \rangle$, $u.T \subseteq \mathcal{A}$ is the text describing the user's desired features, $u.L \in \mathbb{R}^2$ denotes the desired location and $\alpha \in [0, 1]$ denotes the importance of location over matching the desired features. Given a preference u, we can assign

a score to each object using the following equation:

$$f(o,u) = \alpha \times \delta(o.L, u.L) + (1-\alpha) \times \theta(o.T, u.T) \quad (1)$$

where $\delta(o.L, u.L)$ is the spatial distance, and $\theta(o.T, u.T)$ is the textual distance between the object o and the user preference u. Given an integer k, we can return the top-k spatio-textual objects according to their score. In the scope of this paper, we assume that lower scores are better, both spatial and textual distances are normalized in the interval $[0, 1]$ and $f(o,u) = 1.0$ if $\theta(o.T, u.T) = 1$. The latter assumption implies that objects that are not textually relevant to the query cannot be considered as a valid result.

The textual relevance we employ is the normalized intersection of terms between the description of a spatio-textual object $o.T$ and a user preference keyword set $u.T$, i.e., $\theta(o.T, u.T) = 1 - |o.T \bigcap u.T||u.T|^{-1}$. Although in large documents different textual similarity functions are more appropriate, the intersection is more representative in cases of feature selection. For instance if a user is looking for a hotel with a restaurant and a pool, any hotel offering more features (e.g. restaurant, pool, bar) than the ones specified by the user should not be less textually relevant than a hotel which offers only the features specified by the user preference (restaurant, pool).

Definition 1 ***Top-k query.*** *Given a set D of spatio-textual objects, a set of terms $\mathcal{A}$, a scoring function f, an integer k, and a query u, the result set $TOP_k(u)$ of a top-k query is a set of spatio-textual objects such that $TOP_k(u) \subseteq D$, $|TOP_k(u)| = k$ and $\forall o_1, o_2 : o_1 \in TOP_k(u), o_2 \in D - TOP_k(u)$ it holds that $o_1.T \bigcap u.T \neq \emptyset$ and $f(o_1, u) \leq f(o_2, u)$.*

If an object o belongs to the $TOP_k(u)$ set of a user preference u, we say that o is *visible* to u or that u *sees* o. For a specific set of objects D and a set of user preferences U, it is possible to identify for a query object q the set of users who can see q. This is the reverse procedure of a top-k query and therefore it is called *reverse top-k* ($RTOP_k$) *query* [18].

Definition 2 $RTOP_k$ ***query.*** *Given a set D of spatio-textual objects, a set of user queries U, a scoring function f, integer k, and a spatio-textual object q, the result set $RTOP_k(q)$ of a reverse top-k query is set such that $RTOP_k(q) \subseteq U$ and $u \in RTOP_k(q)$ if and only if $\exists o \in TOP_k(u)$ such that $f(q,u) \leq f(o,u)$.*

The cardinality of the $RTOP_k$ set of a query-object q is called *influence score* of the object and we denote it as $I(q)$. The influence score indicates the number of users to whom q is visible.

3.2 IR-tree

We employ a state-of-the-art index structure to process spatial keyword queries, namely the IR-tree [6]. The IR-tree is an R-tree where each node is associated with an inverted index of the objects contained in the respective sub-tree rooted

at the node. The IR-tree offers the possibility of retrieving objects that are near a query point but not textually relevant to it, a property that is essential in identifying possibly interesting terms. In more detail, each leaf node contains an inverted index of the spatio-textual objects contained in the node. The leaf node is characterized by a spatio-textual pseudo-object which consists of a *minimum bounding rectangle* (MBR) that encloses all objects of the node and a pseudo-document that consists of the union of all the terms contained in the children of the node. Each non-leaf node contains an inverted index of the spatio-textual pseudo-objects of the children nodes it contains. Non-leaf nodes are also characterized by spatio-textual pseudo-objects which are constructed similarly to the pseudo-objects of the leaf nodes.

4 Problem Definition

Given a set of spatio-textual objects D and a set of spatio-textual preferences U, the influence score of an object q is the number of preferences to which q is visible. Assuming that the location of a spatio-textual object cannot change, the only the way to improve the influence score of q is to enhance its textual description, in order to increase the textual relevance between q and the user preferences in U. In this paper, we study the problem of finding a set of b terms, which when added to the textual description of q, they maximize the influence score of q. We refer to this problem as *Best-terms*.

Definition 3 ***Best-terms query.*** *Given a set D of spatio-textual objects, a set of terms $\mathcal{A} = \bigcup_{o \in D} o.T$, a set of queries U, a scoring function f, an integer k, a spatio-textual object $q = \langle q.T, q.L \rangle$, and an integer b, the set* BT *is a set of terms such that* $\mathrm{BT} \subseteq \mathcal{A}$, $\mathrm{BT} \bigcap q.T = \emptyset$, $|\mathrm{BT}| \leq b$ *and* $\forall T \subseteq \mathcal{A} - \mathrm{BT}, |T| \leq b$ *it holds that* $I(q_1) \geq I(q_2)$ *where* $q_1 = \langle q.T \bigcup \mathrm{BT}, q.L \rangle$ *and* $q_2 = \langle q.T \bigcup T, q.L \rangle$.

The Best-terms problem is NP-hard. We show that by studying a special case of a Best-terms query, namely the respective decision problem of finding whether there exists a set of terms T with $|T| \leq b$ such that $I(\langle q.T \bigcup T, q.L \rangle) = |U|$.

Problem 1 ***Best-terms (decision problem).*** *Given a set D of spatio-textual objects, a set of terms $\mathcal{A} = \bigcup_{o \in D} o.T$, a set of queries U, a scoring function f, an integer k, and a spatio-textual object $q = \langle q.T, q.L \rangle \in D$, decide if there is a set* BT *such that* $\mathrm{BT} \subseteq \mathcal{A}$, $\mathrm{BT} \bigcap q.T = \emptyset$, $|\mathrm{BT}| \leq b$ *for which it holds that* $I(q_1) = U$ *where* $q_1 = \langle q.T \bigcup \mathrm{BT}, q.L \rangle$

We will show that Problem 1 is NP-complete by reducing the set cover problem in Problem 1 using the restriction technique [11].

Definition 4 ***Set cover problem.*** *Let U be a set of elements (universe) and $\mathcal{T} = \{T_1, \ldots, T_n\}$ be a collection of sets where $\bigcup_{i=1}^{n} T_i = U$. The* set cover problem *decides if there is a subset of $\mathcal{T}$, $\mathcal{T}' \subseteq \mathcal{T}$ of size $|\mathcal{T}| \leq b$ such that $\mathcal{T}'$ is a cover of U.*

Theorem 1 *The decision problem of Best-terms is NP-complete.*

Proof. Let an oracle machine select the BT set for a query object q. We set $p = \langle q.T \bigcup \mathrm{BT}, q.L \rangle$ and by performing a TOP_k query for each user preference we can calculate the $RTOP_k(p)$ set and the influence score $I(p)$ of object p in polynomial time. Therefore the solution can be verified in polynomial time and our problem belongs to the NP class.

We set U a to be a set of users and $D = \{q\}$, where $q.T = \emptyset$. We define a collection $\mathcal{T} = \{T_1, \ldots, T_{|A|}\}$ of sets, one for each term t_i in A where a user u belongs in T_i only if $t_i \in u.T$. If we consider $k = 1$, then, for all users that $q.T \bigcap u.T = \emptyset$ it holds that $q \notin TOP_k(u)$ since q is not relevant to $u.T$. If $q.T \bigcap u.T \neq \emptyset$ then $q \in TOP_k(u)$ as it is the only object. Therefore any selection of a term t_i is equivalent of selecting a subset of T_i of U. The set cover problem is consequently reduced to Problem 1, as it can be seen as a special case of Problem 1. Problem 1 is therefore NP-complete. Best-terms is at least as hard as Problem 1, which leads us to the conclusion that the Best-terms problem is NP-hard.

5 The Best Term First (BTF) Algorithm

Since the Best-terms problem is NP-hard, an exact solution is infeasible, even for medium-sized datasets. Motivated by this observation, in this section we describe a greedy algorithm, termed *Best Term First* (BTF), that provides an approximate solution to the Best-terms problem. BTF operates in an iterative way consisting of b steps, and in each step it adds to the query object the term that induces the highest increase in influence score.

5.1 Algorithmic Description

Algorithm 1 describes the BTF approach in more detail. BTF takes as input an IR-tree index containing the set of spatio-textual objects D, and an IR-tree index containing the set of user preferences U. BTF works in b iterations, and in each iteration the best term (i.e., the term that induces the maximum increase in the influence of q) is selected and added to the terms of the query object.

Initially, BTF creates a pseudo-preference q' defined by q and using $\alpha = 1$, which indicates that q' uses only distance to data objects, not textual similarity, for ranking. The role of q' is to enable traversing the preference dataset solely based on distance to the query object q. This imitates a sorted access to the preferences, yet this is achieved by means of the IR-tree index on U, without having to sort U.

In each iteration, BTF first creates a set C of candidate spatio-textual objects, one for each term that can be added to q. The size of C is equal to $|\mathcal{A} - q.T|$. In lines 9 and 10 the algorithm exploits the sorted access to the preference dataset, in order to avoid processing some top-k queries. More accurately, given the current user preference u, the score of the last retrieved spatio-textual

Algorithm 1: Best Term First (BTF) Algorithm

Input: U:set of users, D: set of objects,
q:query point, b : number of new terms
Output: BT: set of new terms

$C \leftarrow \emptyset$, buffer $\leftarrow \emptyset$
$q' \leftarrow \langle q.T, q.L, 1\rangle$
bestCandidate$\leftarrow q$
for $i = 0; i < b; i++$ **do** // repeat until b new terms have been found
  forall the $t \in \mathcal{A} - q.T$ **do**
    $C \leftarrow C \bigcup \{\langle \text{bestCandidate}.T \bigcup \{t\}, \text{bestCandidate}.L\rangle\}$
  $u \leftarrow$next(U,q')
  while $u \neq null$ **do**
    $\tau \leftarrow \max_{p \in \text{buffer}}(f(p,u))$ // empty buffer in first iter., so we set $\tau \leftarrow \infty$
    if $\exists c \in C : f(c,u) \leq \tau$ **then**
      buffer$\leftarrow \text{TOP}_k(u)$
      $\tau \leftarrow \max_{p \in \text{buffer}}(f(p,u))$
      forall the $c \in C$ **do**
        if $f(c,u) \leq \tau$ **then**
          $I(c) \leftarrow I(c) + 1$
    $u \leftarrow$next(U,q')
  bestCandidate$\leftarrow \operatorname{argmax}_c(I(c))$
BT$\leftarrow$ bestCandidate.T-q.T
return BT

objects is compared with the scores of the candidate objects C, and if no candidate object has a better score than the k-th ranked spatio-textual object, the user preference is ignored (*pruning condition*) as no candidate object can be in its TOP_k set. Otherwise, the top-k query needs to be executed and its TOP_k result set is stored in the buffer. All candidate objects that are no worse than the k-best element of the calculated TOP_k set belong also to the TOP_k set of u and therefore their influence score is increased. When all user preferences have been examined, the object with the highest influence score is selected and a new set of candidate objects is created based on that object. The procedure is repeated b times until an object with b new terms is created. The b terms that were selected constitute the resulting BT set.

Although BTF adopts a greedy technique to select the b terms, the use of sorted access to dataset U together with the pruning condition reduce the number of processed top-k queries, thereby saving computational costs.

5.2 Complexity Analysis

The cost of the BTF algorithm is determined by the cost of selection of each of the b terms. The main factors that affect the cost of term selection are the

construction of set C with cost $O(|\mathcal{A}|)$, and the cost C_{topk} of processing a top-k query which in worst case will be processed $|U|$ times. Thus, the overall complexity of BTF is equal to $C_{BTF} = O(b(|\mathcal{A}| + |U|C_{topk}))$. However, in practice the number of processed top-k queries is much smaller than $|U|$.

6 Graph-Based Term Selection

BTF extends the textual description of a spatio-textual object iteratively, which forces the algorithm to scan the preferences set U multiple times. In this section we present a novel algorithm, named Graph-Based Term Selection (GBTS), which examines the set of preferences only once and creates a graph of terms that provides an estimation of the influence gain any combination of terms may provide.

Essentially, GBTS consists of two separate algorithms. The first algorithm, named Graph Construction (GC), creates a graph connecting the terms which when added to the spatio-textual query object q, they can induce an increase in its influence score. The second algorithm, named Best Subgraph Selection (BSS), traverses the graph in a deliberate manner, in order to identify the sets of terms that will induce the highest increase in the influence score of q.

6.1 Graph Construction Algorithm

Given a set of objects D, a set of user preferences U and a spatio-textual object q, we denote as $\widehat{U}(q)$ the subset of all preferences ($\widehat{U}(q) \subseteq U$) for which q is not visible and at most b terms are needed for q to become visible. The Graph Construction algorithm builds a weighted graph $G = (V, E)$ where each node of the graph represents a candidate term, and the weights on edges indicate the maximum increase in the influence score of q that can be induced, if the respective set of terms is added to q.

In more detail, for each examined user preference u, the algorithm adds to graph G a node for each previously unseen term. The edges connecting the nodes and the weights of the edges are determined by the number of terms λ that need to be added to u for it to be included in $\text{RTOP}_k(q)$. The value of λ is calculated based on Equation 2, where τ is the worst score that q is required to have in order to be in the $TOP_k(u)$ set and derives directly from Equation 1.

$$\tau = \alpha \times \delta(q.L, u.L) + (1 - \alpha) \times \frac{|q.T \bigcap u.T| + \lambda}{|u.T|} \tag{2}$$

- If $\lambda \leq 1$, the algorithm adds a loop edge with weight equal to 1 to each term t that is not contained in q. If the edge already exists, the weight is simply added to the weight of the existing edge.
- In the case that $\lambda > 1$, i.e., more than one terms are necessary for q to be included to $TOP_k(u)$, the procedure is slightly different. Let $T = u.T - q.T = \{t_1, \ldots, t_n\}$ be the terms that are included in u but not in q. For each pair of terms in $u.T - q.T$, the algorithm adds an edge with weight w_e. As before, if an edge already exists, the weight is added to the existing edge.

Algorithm 2: Graph Construction (GC) Algorithm

Input: U:set of users, D: set of objects, q:query point, b : number of new terms
Output: $G = (V, E)$: resulting graph

1: $V \leftarrow \emptyset, E \leftarrow \emptyset$, buffer$\leftarrow \emptyset$,$G \leftarrow (V, E)$ // graph initialization
2: $q' \leftarrow \langle q.T, q.L, 1 \rangle$
3: $u \leftarrow next(U, q')$
4: **while** $u \neq null$ **do**
5: buffer$\leftarrow$ TOP$_k(u)$
6: $\tau \leftarrow \max_{p \in \text{buffer}} (f(p, u))$
7: **if** $f(q, u) > \tau$ **then** // if $q \notin TOP_k(u)$
8: $T \leftarrow u.T - q.T$
9: $V \leftarrow V \bigcup T$
10: $\lambda \leftarrow \max\left(1, \left\lceil \left(1 - \frac{\tau - a\delta(q,u)}{1-a}\right) |u.T| - |q.T \bigcap u.T| \right\rceil\right)$ // from Eq. 2
11: **if** $\lambda = 1$ **then**
12: $E \leftarrow E \bigcup \{e = (t_i, t_i, 1) : t_i \in T\}$
13: **else if** $1 < \lambda \leq b$ **then**
14: $E \leftarrow E \bigcup \left\{ e = \left(t_i, t_j, \frac{2}{\lambda(\lambda-1)}\right) : \forall t_i, t_j \in T \text{ and } t_i \neq t_j \right\}$
15: $u \leftarrow$next(U,q')
16: **return** G

Since we add λ terms that correspond to $\lambda(\lambda-1)/2$ pairs of terms, the weight of each edge w_e is set to $2\left(\lambda(\lambda-1)\right)^{-1}$, which is a normalization that makes the sum of weights added equal to 1. Intuitively, we add a total weight of 1 to each subgraph $G' = (V', E')$ where $V' \subseteq T$ and $|V'| = \lambda$, indicating the potential increase in the influence score of q if the terms contained in G' were added to q.

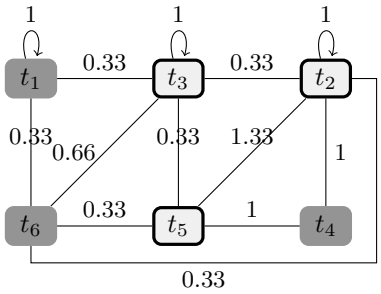
Algorithm 2 describes the construction of the term graph G. Similarly to Algorithm 1, GC traverses the preferences based on their distance to q. For each user preference u, if q is not in the $TOP_k(u)$ set, GC updates the node set of G and calculates λ (line 10), the number of terms that need to be added in q for it to be included in the $TOP_k(u)$ set. A non-positive value of λ indicates that u is located near q but $q.T \bigcap u.T = \emptyset$ and therefore q is not included in the $TOP_k(u)$ set. The addition of any term will allow q to be added to TOP$_k(u)$ set and therefore one loop edge is added to each term t for which it holds $t \in u.T - q.T$. If more than one terms are necessary to be added in q ($\lambda > 1$), GC adds all necessary edges in the graph. The algorithm continues until all user preferences have been examined.

The size of the graph depends on the number of distinct terms contained in $\widehat{U}(q)$. The terms correspond to the features extracted from the textual descriptions of spatio-textual objects that describe the offered facilities. In practice, we have noticed that the vocabulary for the targeted applications is limited and therefore the graph is expected to fit in main memory.

User terms

user	terms	min terms to be added
u_1	t_1, t_2, t_3	1
u_2	t_2, t_4, t_5	2
u_3	t_2, t_3, t_5, t_6	3
u_4	t_1, t_3, t_6	3

(a) The term-sets for the users



(b) The resulting graph

Fig. 1. Example graph: The nodes of the suggested solution are colored with light gray

Example 1 *As an example, let the user preferences in Figure 1(a) be the* $\widehat{U}(q)$ *set for* $b = 3$*, i.e., the set of user preferences that can be added to the* $RTOP_k(q)$ *set if 3 more terms are added to the spatio-textual object* q*. We also assume that the shown terms for each user preference are not included in* q*. The first step of the algorithm is the evaluation of the user preference* u_1*. The algorithm adds to the graph the nodes* t_1, t_2, t_3 *and since only one term needs to be added to* q *for* u_1 *to be added to* $RTOP_k(q)$*, it adds one loop edge with weight 1 to all terms. On the next step* u_2 *is processed and two more nodes* (t_4, t_5) *are added to the graph. For each pair of the terms contained in* u_2 *we add an edge to the graph with weight equal to* $2(\lambda(\lambda-1))^{-1}$ *where* λ *is equal to 2 which the number of terms needed to be added to* q*, for* u *to be in* $RTOP_k(q)$*. When* u_3 *is processed,* t_6 *is added to the graph and for each pair of terms in* u_3 *an edge with weight* $1/3$ *is added to graph. Finally,* u_4 *is processed and the graph is updated accordingly.*

6.2 Best Subgraph Selection Algorithm

When the graph has been created, the Best Subgraph Selection algorithm (BSS) chooses as seed nodes the b nodes (terms) of the graph with the highest degree and creates a set of b subgraphs with initially one node each. Next, each subgraph is expanded by adding at each step the node with highest degree that is adjacent to a node of the subgraph. The expansion of each subgraph is continued until each subgraph has b nodes or the subgraph cannot be expanded. Finally, the subgraph with the highest sum of edge weights is selected as solution and the set of terms included in the subgraph are the ones that constitute the BT set.

Algorithm 3 describes the algorithm of term selection. Initially an empty priority queue (Q) is constructed. Subsequently, at line 3 the algorithm chooses as seed the highest degree node t_i that has not yet been selected and constructs the subgraph G_{t_i} (line 4). The subgraph is constructed by repeatedly selecting the highest degree node adjacent to the G_{t_i} until $|G_{t_i}| = b$ or until no nodes can be added to G_{t_i}. When each subgraph is constructed, it is pushed to Q. The sorting key of Q is the sum of weights of the edges in the subgraph. The BT

Algorithm 3: Best Subgraph Selection (BSS) Algorithm

Input: $G = (V, E)$: graph, b: number of desired terms
Output: BT:set of new terms

```
Q ← ∅, BT ← ∅
for i = 0; i < b; i++ do
    t_i ← next node of G with the highest degree
    G_{t_i} ← createSubgraph(t_i)
    Q.add(sumOfWeights(G_{t_i}),G_{t_i})
while |BT| ≤ b do
    G_S ←Q.pop()
    add to BT the b − |BT| highest degree nodes from G_S
return BT
```

set is constructed by selecting the subgraph with the highest sum of edges and adding the terms of the subgraph to BT. If the subgraphs contain less than b terms, more subgraphs are pulled from the priority queue until BT contains b terms. In such cases we add from each subsequent subgraph to BT the $b - |\text{BT}|$ highest degree nodes of the subgraph.

Example 2 *Continuing the previous example, during the execution of BSS, 3 subgraphs are created with seed nodes the terms t_2, t_5 and t_3. We denote the respective subgraphs as G_{t_i} where t_i is the seed node of the subgraph. Each subgraph G_{t_i} is extended to the highest degree node adjacent to G_{t_i}. In the case of G_{t_2}, the subgraph is expanded by adding first node t_5, which is the node with the highest degree adjacent to t_2 and subsequently with node t_3 which is the highest degree node adjacent to either t_2 or t_5. After the addition of t_3, the size of G_{t_2} becomes equal to 3 and therefore the expansion stops and the next subgraph is processed. In the case of the example all subgraphs produce the same result which includes the light gray nodes in Figure 1(b). The nodes contained in the result are the ones to be added to q.*

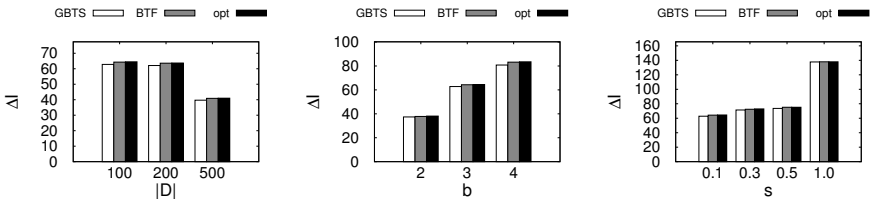
6.3 Complexity Analysis

The overall complexity of Graph-Based Term Selection is determined by the two algorithms that comprise it.

$$C_{GBTS} = C_{GC} + C_{BSS}$$

GC consists of two parts: the processing of $|U|$ top-k queries and the addition of edges $\widehat{U}(q)$ times. The addition of an edge is done in constant time and therefore the cost of GC is equal to: $C_{GC} = O(|U| \cdot C_{topk} + \widehat{U}(q))$.

BSS also consists of two parts: the construction of b subgraphs, and the selection of nodes (terms) from the best of these subgraphs. The main cost of BSS is the construction of the b subgraphs, which is $O(b \cdot (b^3 + logb))$. The cost of expanding a single-node subgraph b times and finding the highest degree node is equal to $O(b^3)$, while the cost of insertion to the priority queue is equal to



(a) Varying $|D|$ (b) Varying b (c) Varying s

Fig. 2. Evaluating the quality of results

$O(logb)$. The node selection is in worst case $O(b \cdot logb)$, though in practice it is $logb$. Hence, we derive: $C_{BSS} = O(b \cdot (b^3 + logb))$.

Consequently, the overall complexity of Graph-Based Term Selection is equal to: $C_{GBTS} = O(|U| \cdot C_{topk} + \widehat{U}(q) + b \cdot (b^3 + logb))$.

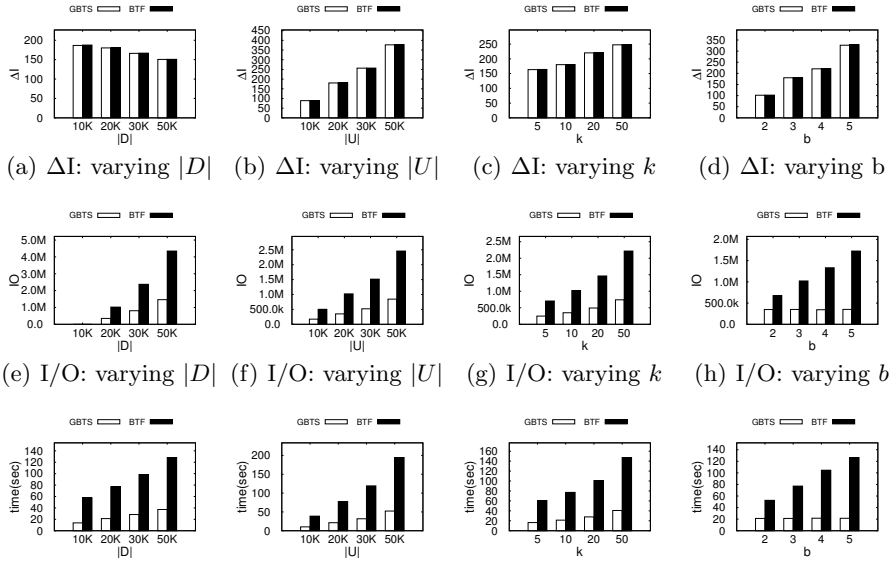
7 Experimental Evaluation

In this section, we present the results of the experimental evaluation. All algorithms were implemented in Java and the experiments were executed on an AMD Opteron 4130 Processor (2.60GHz), with 32GB of RAM and 2TB of disk.

Datasets and metrics. For the data set D of spatio-textual objects, we used a set of 200000 descriptions of hotels from the site of Booking.com[1]. The dataset contains 188 distinct features. The set of preferences U was generated using a uniform distribution for creating the location and the α parameter of each preference, while the terms were randomly chosen from the vocabulary generated by processing the set of hotels. The location of the user preferences was bounded in the MBR defined by set of hotels. We also tested our algorithm against a Zipfian distribution of terms. We used the Zipfian distribution generator provided by the Apache Commons project[2]. The metrics under which we evaluated the implemented algorithms were: a) increase in the influence score ΔI, b) number of I/O's performed by each algorithm, and c) processing time.

Experimental procedure. Both datasets D and U were indexed using an IR-tree where the maximum capacity of each node was 100 entries. We employed a buffer which was fixed at the size of 4MB, for both the tree index and the inverted files. The performance of the proposed algorithms was evaluated through a series of experiments varying the parameters of a) the cardinality of D in the interval [10K,200K], b) the cardinality of U, [10K,200K], c) the number of returned results per user preference k, [5,50], d) the maximum size of user preferences, [1,5], and d) the number of returned terms for a query object b, [2-5]. For the Zipfian distribution we varied the value of the characteristic exponent s in the interval [0.1-1.0]. The default setup for the experiments was: $|D| = 20K$,

[1] http://www.booking.com
[2] http://commons.apache.org/proper/commons-math/



(a) ΔI: varying $|D|$ (b) ΔI: varying $|U|$ (c) ΔI: varying k (d) ΔI: varying b

(e) I/O: varying $|D|$ (f) I/O: varying $|U|$ (g) I/O: varying k (h) I/O: varying b

(i) time:varying $|D|$ (j) time:varying $|U|$ (k) time:varying k (l) time:varying b

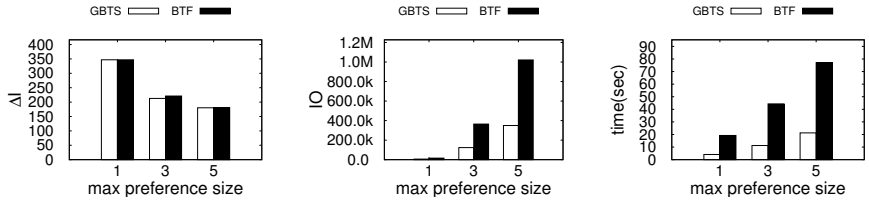
Fig. 3. Analysis for varying $|D|, |U|, k$, and b

$|U| = 20K$, $k = 10$, $b = 3$ and each the maximum preference size was set to 5. For each experiment a random set of 20 query objects was selected from D.

Quality evaluation. We compared the proposed algorithms against an exhaustive algorithm, which examines all $\binom{|A-q.T|}{b}$ term combinations[3] and calculates the optimal set of terms BT. Due to the high processing cost of the exhaustive algorithm even for small values of b, we employed datasets of limited size. The default setting for this series of experiments was $|D| = 100, |U| = 1000$, and $b = 3$. The set of objects D, consisted of a random set of hotels from the area of Catalonia in Spain, and they were selected from Booking.com. The set of preferences U, follows a uniform distribution in Figures 2(a) and 2(b), and a Zipfian distribution in Figure 2(c). Figure 2 indicates that both algorithms achieve an increase to the influence score which is very close to the optimal value. The execution time of the exhaustive algorithm was in all cases orders of magnitude larger than the execution time of BTF and GBTS.

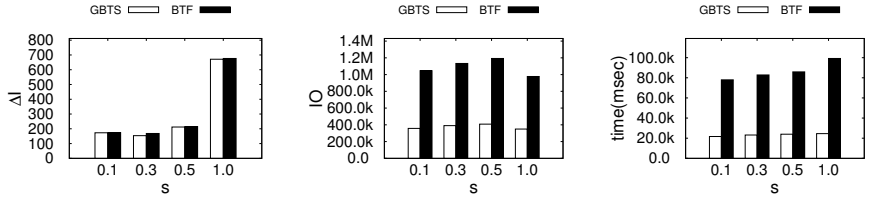
Varying $|D|$. Figures 3(a), 3(e), and 3(i) illustrate the performance of the algorithms as we vary the number of spatio-textual objects. Figure 3(a) indicates that both algorithms perform similarly with respect to the increase of the influence score. As the number of objects increase the gain in influence score drops as more spatio-textual objects compete for the same number of user-preferences and therefore it becomes harder for a query object to increase its influence score.

[3] Based on the adopted similarity function, the addition of a term does not have a negative effect on the influence score. In the general case, an exact algorithm should examine $2^{|A|}$ term combinations.



(a) ΔI (b) I/O (c) time

Fig. 4. Varying max preference size



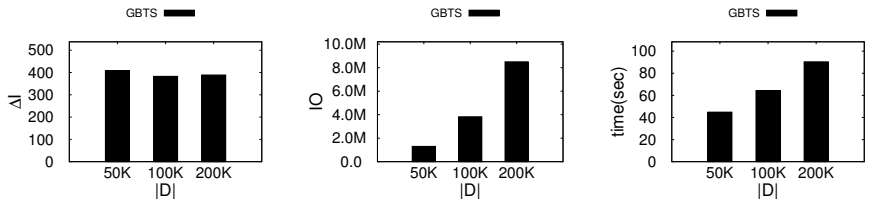
(a) ΔI (b) I/O (c) time

Fig. 5. Varying zipf distribution

Figures 3(e) and 3(i) indicate that the I/O accesses and the processing time for both algorithms increase when the dataset size increases. As the dataset size increases the cost of a single TOP_k query increases as well and therefore both algorithms are affected by the dataset size. The effect on BTF is larger than in GBTS as BTF accesses the data multiple times in order to create the set of new terms.

Varying $|U|$. Figures 3(b), 3(f), and 3(j) depict the performance of both algorithms as more preferences are processed. When the number of preferences increases there are more user preferences that can be added to the RTOP_k set of an object with an addition of a new set of terms and therefore the gain in influence score increases as well. The processing cost for both algorithms is expected to raise for a larger number of user preferences, as more preferences have be to examined. Both processing time and I/O cost raise faster for BTF than for GBTS. In particular the processing cost for BTF grows almost by a factor of b faster than GBTS as BTF has to process the set of preferences b times in order to identify the set of new terms.

Varying k. As the size of the TOP_k set of each preference increases, the cost of a single TOP_k query increases as well. Figures 3(c), 3(g), and 3(k) indicate that the increased I/O and processing cost of a TOP_k query affects both algorithms. Similarly to the increase on the size of datasets, the effect on BTF is magnified by a factor of b. The influence score gain raises as well, since with the increase in k more objects can be included in the TOP_k set of a user preference and the necessary increase in the text similarity for a query object q to be added to a TOP_k set of a user preference u becomes smaller.



(a) ΔI (b) I/O (c) time

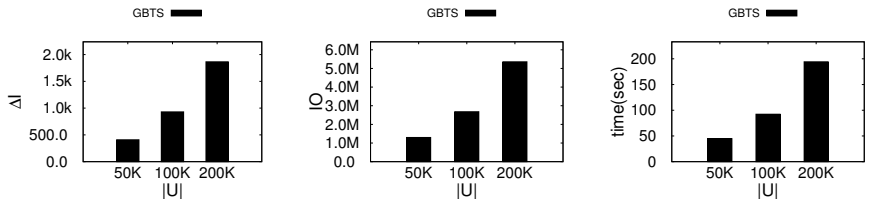
Fig. 6. Varying data cardinality

Varying b. Figures 3(d), 3(h), and 3(l) illustrate the performance of the algorithms as we vary the number of new terms added to each query object. It is noteworthy that both algorithms behave similarly with respect to the increase of the influence score. The cost of BTF raises linearly with respect to b, which is expected as it has to process the data b times before returning the resulting BT set. On the other hand, GBTS remains unaffected by the increase of the b parameter, as it has to access the preferences set only once.

Varying the query size. Figure 4 indicates that as the maximum preference size increases, the possible gain of influence score for a spatio-textual object drops. The reason lies in the fact that for a large user preference u, more terms are required to be added to a spatio-textual object q, for q to enter the $TOP_k(u)$ set. Larger queries require more complex TOP_k queries on the indexes and consequently the performance of both algorithms is affected. As expected BTF is affected in a larger degree than GBTS by the increased cost of the TOP_k queries.

Zipfian distribution. It is quite common that the terms of user-preferences follow a Zipfian distribution. We tested our algorithms against a set of user preferences where the occurrences of terms follow a Zipfian distribution. Figure 5 illustrates the experimental results. Similarly to the uniform distribution, GBTS outperforms BTF in terms of I/O accesses and processing time while producing the same gain in influence score. In cases where the exponent of the Zipfian distribution takes high values the gain in influence score raises significantly. Such behavior is expected as when a small number of distinct terms appear in a large number of user preferences, adding those terms to a spatio-textual object will result in a significant increase of its influence score since the addition of those terms will allow it to enter the TOP_k set of many user preferences.

Scalability analysis. We evaluated the performance of GBTS against larger datasets to evaluate the scalability of our approach. BTF is not included in the results as it needed excessive time to produce results. The experimental results shown in Figure 6 indicate that the processing time of GBTS grows logarithmically with respect to the size of the D, while the I/O cost increases linearly. The performance difference between processing time and I/O accesses lies in the fact that in the first TOP_k queries we have an increased number of I/Os, however after a certain number of queries, several nodes of the IR-tree are buffered and as a result the subsequent TOP_k queries induce a limited number



(a) ΔI (b) I/O (c) time

Fig. 7. Varying preferences cardinality

of I/O accesses. Figure 7 illustrates the performance of GBTS with respect to the cardinality of user preferences set. Both the processing time and the I/O increase linearly with respect to time.

8 Conclusions

In this paper, we address the challenging problem of increasing the influence of a spatio-textual object, by enriching its textual description with at most b carefully selected keywords. In this way, the spatio-textual object's textual relevance to user queries is increased, with the ultimate objective being for the object to become part of the top-k result for many different users. We provide a formal problem statement that is novel and relies on concepts related to top-k and reverse top-k queries. We show that the problem is NP-hard, and we present a greedy solution to the problem. Then, we propose a more efficient algorithm that achieves results of comparable quality, but with significantly lower processing cost. We demonstrate the performance gains of the proposed approach by means of a thorough experimental evaluation that includes real data.

Acknowledgments

A. Vlachou was supported by the Action "Supporting Postdoctoral Researchers" of the Operational Program "Education and Lifelong Learning" (Action's Beneficiary: General Secretariat for Research and Technology), and is co-financed by the European Social Fund (ESF) and the Greek State. C. Doulkeridis has been co-financed by ESF and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Aristeia II, Project: ROADRUNNER.

References

1. R. Agrawal, A. Gupta, Y. Prabhu, and M. Varma. Multi-label learning with millions of labels: recommending advertiser bid phrases for web pages. In *Proc. of WWW*, pages 13–24, 2013.

2. X. Cao, G. Cong, and C. S. Jensen. Retrieving top-k prestige-based relevant spatial web objects. *PVLDB*, 3(1):373–384, 2010.
3. L. Chen, G. Cong, C. S. Jensen, and D. Wu. Spatial keyword query processing: An experimental evaluation. *PVLDB*, 6(3):217–228, 2013.
4. Y. Choi, M. Fontoura, E. Gabrilovich, V. Josifovski, M. R. Mediano, and B. Pang. Using landing pages for sponsored search ad selection. In *Proc. of WWW*, pages 251–260, 2010.
5. S. Cholette, Ö. Özlük, and M. Parlar. Optimal keyword bids in search-based advertising with stochastic advertisement positions. *J. Optimization Theory and Applications*, 152(1):225–244, 2012.
6. G. Cong, C. S. Jensen, and D. Wu. Efficient retrieval of the top-k most relevant spatial web objects. *PVLDB*, 2(1):337–348, 2009.
7. K. Deng, X. Li, J. Lu, and X. Zhou. Best keyword cover search. *IEEE Trans. Knowl. Data Eng.*, 27(1):61–73, 2015.
8. R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. *J. Comput. Syst. Sci.*, 66(4):614–656, 2003.
9. A. Fuxman, P. Tsaparas, K. Achan, and R. Agrawal. Using the wisdom of the crowds for keyword generation. In *Proc. of WWW*, pages 61–70, 2008.
10. Y. Gao, X. Qin, B. Zheng, and G. Chen. Efficient reverse top-k boolean spatial keyword queries on road networks. *IEEE Trans. Knowl. Data Eng.*, 27(5):1205–1218, 2015.
11. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Macmillan Higher Education, 1979.
12. X. Lin, J. Xu, and H. Hu. Reverse keyword search for spatio-textual top-k queries in location-based services. To appear in *IEEE Trans. Knowl. Data Eng.*, 2015.
13. J. Lu, Y. Lu, and G. Cong. Reverse spatial and textual k nearest neighbor search. In *Proc. of SIGMOD*, pages 349–360, 2011.
14. Y. Lu, J. Lu, G. Cong, W. Wu, and C. Shahabi. Efficient algorithms and cost models for reverse spatial-keyword k-nearest neighbor search. *ACM Trans. Database Syst.*, 39(2):13, 2014.
15. P. Papadimitriou, H. Garcia-Molina, A. Dasdan, and S. Kolay. Output URL bidding. *PVLDB*, 4(3):161–172, 2010.
16. S. Ravi, A. Z. Broder, E. Gabrilovich, V. Josifovski, S. Pandey, and B. Pang. Automatic generation of bid phrases for online advertising. In *Proc. of WSDM*, pages 341–350, 2010.
17. J. B. Rocha-Junior, O. Gkorgkas, S. Jonassen, and K. Nørvåg. Efficient processing of top-k spatial keyword queries. In D. Pfoser, Y. Tao, K. Mouratidis, M. A. Nascimento, M. F. Mokbel, S. Shekhar, and Y. Huang, editors, *SSTD*, volume 6849 of *Lecture Notes in Computer Science*, pages 205–222. Springer, 2011.
18. A. Vlachou, C. Doulkeridis, Y. Kotidis, and K. Nørvåg. Reverse top-k queries. In *Proc. of ICDE*, pages 365–376, 2010.
19. A. Vlachou, C. Doulkeridis, K. Nørvåg, and Y. Kotidis. Branch-and-bound algorithm for reverse top-k queries. In *Proc. SIGMOD*, pages 481–492, 2013.
20. D. Wu, M. L. Yiu, G. Cong, and C. S. Jensen. Joint top-k spatial keyword query processing. *IEEE Trans. Knowl. Data Eng.*, 24(10):1889–1903, 2012.
21. D. Zhang, C. Chan, and K. Tan. Processing spatial keyword query as a top-k aggregation query. In *Proc. of SIGIR*, pages 355–364, 2014.
22. D. Zhang, K. Tan, and A. K. H. Tung. Scalable top-k spatial keyword search. In *Proc. of EDBT*, pages 359–370, 2013.
23. W. Zhang, D. Wang, G.-R. Xue, and H. Zha. Advertising keywords recommendation for short-text web pages using Wikipedia. *ACM TIST*, 3(2):36, 2012.