Interactive Top-k Spatial Keyword Queries

Kai Zheng[1], Han Su[1], Bolong Zheng[1], Shuo Shang[2], Jiajie Xu[3], Jiajun Liu[4], Xiaofang Zhou[1,3]

[1] *School of Information Technology and Electrical Engineering,*
The University of Queensland, Brisbane, Australia
{uqkzheng, h.su1, b.zheng, zxf@uq.edu.au}

[2] *China University of Petroleum, Beijing*
sshang@cup.edu.cn

[3] *School of Computer Science, Soochow University, China*
{xujj,zxf}@suda.edu.cn

[4] *AS Program, CSIRO, Pullenvale, Australia*
Jiajun.Liu@csiro.au

***Abstract*—Conventional top-k spatial keyword queries require users to explicitly specify their preferences between spatial proximity and keyword relevance. In this work we investigate how to eliminate this requirement by enhancing the conventional queries with interaction, resulting in Interactive Top-k Spatial Keyword (ITkSK) query. Having confirmed the feasibility by theoretical analysis, we propose a three-phase solution focusing on both effectiveness and efficiency. The first phase substantially narrows down the search space for subsequent phases by efficiently retrieving a set of geo-textual k-skyband objects as the initial candidates. In the second phase three practical strategies for selecting a subset of candidates are developed with the aim of maximizing the expected benefit for learning user preferences at each round of interaction. Finally we discuss how to determine the termination condition automatically and estimate the preference based on the user's feedback. Empirical study based on real PoI datasets verifies our theoretical observation that the quality of top-k results in spatial keyword queries can be greatly improved through only a few rounds of interactions.**

I. Introduction

With the rapid transformation of web clients from desktop computers to mobile devices such as smartphones and tablets, increasing volumes of geo-textual objects are becoming available on the web that represent Points of Interest (PoIs) such as restaurants, cafes and hotels. Specifically, a geo-textual object contains the geo-location (usually in the form of longitude and latitude) of its PoI and textual descriptions of the PoI (e.g., features, facilities, reviews). There are now numerous online sources, from which large-scale geo-textual objects can be acquired, including business directories such as Google Places for Business[1] and Yahoo!Local[2], location-bases social networks such as Foursquare[3], as well as rating and review services such as TripAdvisor[4] and Dianping[5]. This calls for techniques to support the efficient processing of *spatial keyword queries*, which take a geo-location and a set of keywords as arguments and return relevant PoIs that matches the arguments. According to [1][2], these queries can be categorized as follows based on their way of specifying spatial and textual predicates.

- **Boolean Range Queries** [3] retrieve all objects whose text description contains *all* the query keywords and whose location is within the query region.
- **Boolean kNN Queries** [4][5] retrieve the k objects nearest to the query location and each object's text description contains *all* the query keywords.
- **Top-k Range Queries** retrieve up to k objects whose location is within the query region and has the highest textual relevance to the query keywords.
- **Top-k kNN Queries** [6][7] retrieve the k objects with the highest ranking scores, measured by a weighted combination of their distances to the query location and the textual similarity between their textual descriptions and query keywords.

We summarise the major characteristics of each query type in Table I. Generally, a new query type is proposed in order to improve previous queries. For instance, the result set of Boolean Range Queries has uncontrolled size and is unranked, leading to too many/few results. Boolean kNN Queries address this problem by applying a rank over the results according to their distances to the query location and returning the k closest objects only. However, both types of queries require each result to contain all the query keywords, which may find too few results and/or the results are far away from the query location. Top-k Range Queries and Top-k kNN Queries relax this requirement by introducing textual relevance function as the similarity measure between query keywords and text description of PoIs. Top-k Range Queries rank the result set by considering their textual similarity to the query only, while Top-k kNN Queries combine the similarities over spatial and textual dimensions together into a unified utility function and return the top-k results based on this utility function. To some extent, Top-k kNN Queries are the most novel and advanced type of spatial keyword queries in literature, which are often referred to as Top-k Spatial Keyword Queries (TkSK) when the context is clear.

Despite the high flexibility and expressiveness, TkSK queries are faced with two issues regarding to the query

[1]http://www.google.com.au/business/placesforbusiness/
[2]https://local.yahoo.com/
[3]https://foursquare.com/
[4]http://www.tripadvisor.com/
[5]http://www.dianping.com/

TABLE I: Characteristics of different types of spatial keyword queries

Query type	Matching all query keywords required	Controlled result size	Results are ranked	Preferences on spatial and textual dimensions are considered	Preferences on different query keywords are considered
Boolean Range Query	yes	no	no	no	no
Boolean kNN Query	yes	yes	yes	no	no
Top-k Range Query	no	yes	yes	no	no
Top-k kNN Query	no	yes	yes	yes, but users need to specify their preferences explicitly	no
Interactive Top-k Spatial Keyword Query	no	yes	yes	yes and the preferences are learnt from user feedback	yes and the preferences are learnt from user feedback

practicality and result intuitiveness, as explained in below.

(1) It is impractical to ask users to specify their preferences. As mentioned before, TkSK queries combine spatial similarity and textual similarity into one utility function in the form of $\beta S_{spatial}+(1-\beta)S_{text}$, in which $S_{spatial}, S_{text}$ are spatial and textual similarity between query and object respectively and $\beta \in [0,1]$ is a weighting parameter indicating user's preference over spatial and textual dimensions. A high β favours the objects that are geographically closer to the query location while a small β tends to return the objects whose text description is more relevant to the query keywords. Nevertheless the value of β needs to be specified by the user a priori, which can be quite impractical in real applications. In fact, user preferences are often latent and thus hard to be quantified exactly and explicitly.

(2) The results of TkSK queries with respect to the textual similarity may not be as intuitive as the boolean keyword queries. More specifically, consider the well-known vector space model [8] that has been adopted in existing TkSK queries [7]. It calculates the weight for each common keyword of query's and object's text using TF-IDF measure and computes the normalized dot product between vectors of query keywords and object keywords. Though mathematically sound, the results derived from this model may not be desired by the users, since a user would like to assign a larger weight to some keyword simply because she feels it is more important rather than it occurs less frequently in the entire dataset.

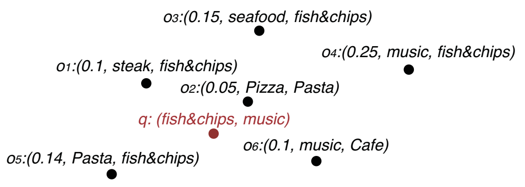

o3:(0.15, seafood, fish&chips)
o4:(0.25, music, fish&chips)
o1:(0.1, steak, fish&chips)
o2:(0.05, Pizza, Pasta)
q: (fish&chips, music)
o6:(0.1, music, Cafe)
o5:(0.14, Pasta, fish&chips)

Fig. 1: An example of spatial keyword query

We use Figure 1 as an example to demonstrate the above two issues. Consider a user who is looking for a Cafe nearby, which must serve fish&chips (more important) and ideally plays music (less important). She issues a spatial keyword query q with her current location and two keywords *fish&ships, music*, as shown in Figure 1. o_1 to o_6 are restaurants/Cafes nearby q with the keywords shown in the parentheses, wherein the number indicates the normalized distance to q. It is not uncommon the user has no idea about how to specify the weight β in the TkSK query [7], so she just accepts the default value $\beta = 0.5$. Since the weight for each keyword is assigned based on TF-IDF model in TkSK query, *music* has much higher weight (= 0.5) than *fish&chips* (=0.25). After a simple calculation, o_6 turns out to be the best object. However this is obviously not a satisfactory result for the user: o_1 is equally close to q with o_6, but contains a more important keyword *fish&chips*, so o_6 is at least worse than o_1.

This work aims to address the above limitations by enhancing spatial keyword queries with user interactions. We assume that the database system interacts with the user in rounds and in each round when presented with a set of geo-textual objects the user can pick the object that she prefers the most. The only arguments needed for this query q are the query location $q.\rho$, a set of query keywords $q.\psi$ and the desired output size $q.k$. Instead of asking the user for her preferences on spatial dimension and different keywords, we learn them automatically from the user's feedback. Our proposed query will have the following desirable features: (1) the importance of each keyword can be distinguished based on user's personal preference; and (2) all the preference weights on spatial proximity and each keyword will be learnt automatically based on user's feedback. Continuing the example in Figure 1, if we present o_1 and o_6 to the user and she picks o_1, then it becomes obvious that she prefers *fish&chips* than *music*.

It is worth mentioning that our work shares some similarities with [9] in terms of the interaction style, which studies the problem of minimizing the regret ratio when the system is enhanced with interactions. Informally speaking, the regret ratio reflects the gap between the maximum utility a user can obtain from the returned k results and from the whole database. It is shown that by carefully presenting k tuples and analysing the user's feedback at each round, the regret ratio can be reduced to an arbitrarily small value. Although their theoretical findings of [9] are promising, its proposed methodologies cannot be applied to our problem due to the following reasons.

First, though it is claimed in [9] that "genuineness" is important for interactive queries, in their work there is only one tuple at each round that "genuinely" exists in the database. Though at the end the system can still retrieve the actual tuple from the database that maximizes the utility function, a user may feel confused or even frustrating during the course of interaction. Consider a location recommendation system that is a typical application of spatial keyword queries. Displaying a lot of faked PoIs may give the users the feeling of unreliability or even fraud. In our proposed system, all the results presented to the users including intermediate and final results are genuine.

Second, the algorithm proposed in [9] (i.e., Algorithm 2) for creating the $k-1$ virtual tuples is relying on the assumption that each attribute of the tuple is numerical, since it will adjust the value of a particular attribute slightly while keeping the other attributes unchanged. However, in geo-textual dataset only the spatial dimension is numerical. It is not clear how the algorithm of [9] can deal with textual attributes.

In our proposal, all the intermediate results presented to users during interaction are real tuples existing in the database. Compared to the manipulation based method [9], we are essentially dealing with a more challenging search problem: ***efficiently search for a subset of tuples in the database at each round that can effectively learn the user's preference based on her choice***. We propose an end-to-end solution that includes practical and efficient algorithms to address these challenges. Our contributions are summarized in below.

- We identify limitations of existing spatial keyword queries. Based on this, we define a novel Interactive Top-k Spatial Keyword (ITkSK) query, which not just allows users to control the preference weights on distance and individual keyword in an intuition-consistent way, but also eliminates the hassles to specify all these parameters explicitly by involving interactions with the users.
- We propose a three-phase solution to process ITkSK query. The first phase (candidate generation phase) quickly narrows the search space from the entire database down to a set of candidates by retrieving *Geo-textual k-skyband set* from the database with respect to the query. In the second phase (interaction phase) we develop several strategies to select a subset of candidates and present them to users at each round, with the aim of maximizing the benefit of learning from the user's feedback. In the last phase (finalisation phase) we discuss how to terminate the interaction automatically and estimate the final preference vector based on a set of linear constraints.
- We conduct empirical study based on real PoI datasets. The favourable results verifies our expectation that ITkSK queries indeed return more satisfactory results by learning a more accurate user preference. Moreover, our interaction strategies are shown to be quite effective in terms of convergence speed.

The remainder of this paper is organized as follows. Section II gives a formal definition of ITkSK query and overviews the solution. Section III,IV and V discuss the technical details of the three phases. Our experimental observations are presented in Section VI, followed by a brief discussion of related work in Section VII. Section VIII concludes this paper.

II. Problem Statement

This section formally defines the ITkSK query and outlines the proposed solution. Some major notations used throughout the paper are listed in Table II.

A. Preliminaries

Definition 1 (Geo-textual object)**.** *Let D be a geo-textual dataset. Each geo-textual object $o \in D$ is defined as a pair $(o.\rho, o.\psi)$, where $o.\rho$ is a 2-dimensional geographical location with longitude and latitude and $o.\psi$ is a text document represented by a set of keywords or terms.*

TABLE II: Summary of notations

Notation	Definition
D	A database of geo-textual objects
q	A spatial keyword query
o	A geo-textual object
$o.\rho$	Geographical location of o
$o.\psi$	A set of keywords associated with o
$\mathbf{w}$	A user preference vector
$\mathbf{w}'$	An estimated user preference vector
$u_{q,w}(o)$	A utility function evaluating the utility score of o w.r.t. q and $\mathbf{w}$
k	The number of final results
κ	The maximum number of intermediate results displayed at each round
S	The candidate set
R	The selected candidate to present to users at each round
L^{α}	The set of constraints obtained in round α
P	A polytope in multi-dimensional space

Definition 2 (Utility function)**.** *Let $(q.\rho, q.\psi = \{t_1, t_2, ..., t_m\})$ be a spatial keyword query specified by a user, $\mathbf{w} = \{w_0, w_1, w_2, ..., w_m\}$ be a vector representing user preference, in which $\forall i \in [0, m], 0 \leq w_i \leq 1$. For each geo-textual object $o \in D$, the user's utility obtained from o can be evaluated by the following utility function,*

$$u_{q,w}(o) = w_0(1 - d(q.\rho, o.\rho)) + \sum_{i=1}^{m} w_i h_o(q.t_i) \qquad (1)$$

where $d(q.\rho, o.\rho)$ is a function that normalizes the Euclidean distance between o and q into the range $[0, 1]$ and h is a function indicating the existence of t_i in o, i.e.,

$$h_o(q.t_i) = \begin{cases} 1, \text{if } q.t_i \in o.\psi \\ 0, \text{otherwise} \end{cases}$$

When context about q and $\mathbf{w}$ is clear, we simply use $u(o)$ to represent $u_{q,w}(o)$.

Compared to the scoring functions adopted in previous work of top-k spatial keyword search such as [7], this utility function has two advantages: (1) finer-grained preferences can be specified on each keyword in the query; and (2) the weight of each keyword intuitively reflects the user's preference on it.

Definition 3 (Top-k Spatial Keyword Query)**.** *Given a geo-textual dataset D, a query $q : (q.\rho, q.\psi)$, a preference vector $\mathbf{w}$ and the number of results k, a top-k spatial keyword (TkSK) query returns a set S of up to k objects from D such that they have the highest utilities w.r.t. $u(o)$, i.e.,*

$$S = \{S \subseteq D, |S| = k | \forall o \in S, o' \in D \setminus S, u(o) \geq u(o')\}$$

If the preference vector $\mathbf{w}$ is specified by the user explicitly, this query can be answered efficiently by utilising existing hybrid indexing structures and query processing algorithms [2]. However, it is often impractical to require a non-expert user to provide the exact value for each element of $\mathbf{w}$. Even worse, sometimes a user may be unsure of her preference before exploring some tuples in the database. In other words, the user preferences in real applications are usually latent. This motivates us to automatically infer the user preferences by involving interactions during query processing. We formally state the problem to be studied in this paper as follows.

Definition 4 (Interactive Top-k Spatial Keyword Query)**.** *Given a geo-textual dataset D, a query $q : (q.\rho, q.\psi)$, an* *integer k and an unknown preference vector $\mathbf{w}$, the interactive top-k spatial keyword (ITkSK) query will be processed in rounds. In each round, the system displays at most κ tuples to the user and asks her to pick the favourite one according to $\mathbf{w}$. After interaction, the system will estimate the user's preference as $\mathbf{w}'$ based on her feedbacks and return a final set of k tuples based on $\mathbf{w}'$.*

In this above definition κ is a predefined constant in order to limit the number of objects displayed to the user. The value of κ can be specified by HCI experts, determined based on psychological study (to make users feel conformable) or customized by end users. In this work we assume that the user's preference remains unchanged during the interaction process. That is, the object o selected by the user at each round must be the one with the highest utility according to $u_{q,w}(o)$ among the k tuples presented at the same round. We leave the scenarios where the user's preference may shift as she sees more tuples in the database as an interesting problem to be investigated in the future.

B. Solution Overview

The proposed solution consists of three phases:

1) **Candidate generation:** This phase will find a set of geo-textual k-skyband tuples from the entire database as the initial candidates.
2) **Interaction:** This phase proceeds in the fashion of rounds. At each round, the system strategically selects a subset of candidates and presents them to the user, who will pick her favourite tuple according to her latent preference. The system will refine the candidate set based on her feedback and continue the process by selecting another subset of candidates. Meanwhile, a termination condition is checked automatically and once satisfied the system exits this phase.
3) **Finalisation:** This phase estimates the user's preference $\mathbf{w}'$ based on her feedbacks during the interaction phase and retrieve the top$k$ results based on $\mathbf{w}'$.

III. Candidate Generation

There usually exists a large number of geo-textual objects in databases. Therefore it is quite inefficient to search for intermediate results through the entire database at each round during the interaction process. The purpose of this phase is to reduce the search space by generating a smaller candidate set.

A. Geo-textual dominance

Since the user preferences on spatial distance and keywords are unknown at this stage, the desired candidate set should include all the objects that could possibly become a final result given a certain preference vector. Naturally we can adopt the notion of skyline [10], which is a set of tuples in a database that are not *dominated* by any other tuple. Here a tuple a is said to *dominate* tuple b if a has better or equal values in all attributes and a better value in at least one attribute than b. In the sequel, we first define the dominance relationship between two geo-textual objects.

Definition 5 (Dominance)**.** *Let a and b be two geo-textual objects in D. Given a spatial keyword query q, a dominates b w.r.t. q, denoted by $a \prec_q b$, if (i) $d(q.\rho, a.\rho) \leq d(q.\rho, b.\rho)$, $q.\psi \cap a.\psi \supset q.\psi \cap b.\psi$; or (ii) $d(q.\rho, a.\rho) \leq d(q.\rho, b.\rho)$, $q.\psi \cap a.\psi \supseteq q.\psi \cap b.\psi$. Otherwise a does not dominate b, denoted as $a \not\prec_q b$. Whenever context of q is clear, we simply write $a \prec b$ ($a \not\prec b$).*

Based on the dominance relationship, we can define geo-textual skyline in below.

Definition 6 (Geo-textual Skyline)**.** *Given a spatial keyword query q, an object $o \in D$ is a geo-textual skyline tuple w.r.t. q if and only if $\forall o' \in D, o' \not\prec_q o$.*

For a given D and q, the geo-textual skyline is guaranteed to contain the top-1 (best) result w.r.t. the utility function $u_{q,w}$ for any preference vector $\mathbf{w}$. In order to guarantee the top-k candidates, we extend skyline to k-skyband [11].

Definition 7 (Geo-textual k-Skyband)**.** *Given a spatial keyword query q, an object $o \in D$ is a geo-textual k-skyband tuple w.r.t. q if and only if o is dominated by at most k tuples in D. We denote the set of tuples forming the k-skyband of D as $SB_q^k(D)$.*

Lemma 1. *$\forall o \in D \setminus SB_q^{k-1}(D)$, o cannot belong to the top-k results w.r.t. $u_{q,w}$ for any preference vector $\mathbf{w}$.*

Proof: By definition of skyband, if $o \notin SB_q^{k-1}(D)$, then o is dominated by at least k tuples in D, which means there exist at least k tuples o' such that $u(o') > u(o)$ for any $\mathbf{w}$. ■

B. Search Algorithm

Following the branch-and-bound paradigm [11], next we propose the GSB (Geo-textual SkyBand) algorithm to find the candidate set efficiently.

IR^2-Tree Index: GSB makes use of an IR^2-Tree that has been widely adopted to facilitate top-k spatial keyword queries. Basically an IR^2-Tree is a combination of R-Tree and signature files, where each node contains two types of information: (1) the minimum bounding area of its subtree; (2) signature of the node, which is the superimposition (OR-ing) of all the signatures of its entries. A signature of a word is fixed-length bit string generated by using a hash function and a signature of a keyword set simply superimposes the hash values of all keywords. A nice property of signature is that, given a query signature s_a and node signature s_n, if $s_a = s_a \wedge s_n$, then the node *may* contain some query keywords; otherwise, the query keywords do not exist in the node. Readers can refer to [5] for more implementation details of IR^2-Tree.

GSB Algorithm: Algorithm 1 illustrates the basic process of GSB search. First GSB initializes a list S that will contain skyband tuples and an empty heap H to hold entries (either a node or a point) in IR^2-Tree to be visited. The heap is sorted in ascending order according to the *minimum geo-textual distance* (MINGTD) of an entry w.r.t. the query q. Let e be the entry with signature s_e and MBR M_e, MINGTD of e is defined by the following function

$$MINGTD_q(e) = MINDIST(q.\rho, M_e) + \sum_{t \in q.\psi} (s_t \wedge s_e) \oplus s_t \quad (2)$$

where MINDIST is the normalized minimum distance between a point and a MBR [12]. The second term compares each query keyword t against the signature of the entry and returns (i) zero if t may be contained in the entry; (ii) one otherwise. Essentially it measures the minimum textual distance between the query and the entry.

Then GSB iteratively removes the top entry e from H and then performs the following actions depending on the entry type, until H becomes empty.

- If e is a non-leaf node, GSB performs a dominance check for each of its child entries to see if it is dominated by more than $k-1$ skyband tuples found so far. An entry e is dominated by a skyband tuple o w.r.t. q if the following conditions are both satisfied:
    - $d(o.\rho, q.\rho) < MINDIST(q.\rho, M_e)$;
    - $\forall t \in q.\psi \setminus o.\psi$, $(s_t \wedge s_e) \neq s_t$

  That is, all the objects in e are farther away from q than o,and do not contain any keyword t in q that is not covered by o. If a sub-entry passes the dominance check, it is added to H; otherwise it is discarded immediately.
- If e is a leaf node, then it contains geo-textual objects only. We perform the dominance check for each object based on Definition 5 using the exact location and keyword information (instead of signature). An object is added to S if it is dominated by less than k tuples in S.

If the heap is empty, the above process terminates. All the objects added to S during the process are guaranteed to be the geo-textual $(k-1)$-skyband tuples, as formally stated by the following theorem.

Theorem 1 (Correctness)**.** *All objects added to S are geo-textual $(k-1)$-skyband tuples.*

Proof: Assume to the contrary that an object o' was added to S but does not belong to the $(k-1)$-skyband. This implies there are at least k objects $o_1, o_2, ..., o_k$ dominating o'. Then according to the nature of GSB algorithm and the evaluation function of MINGTD in Eq. (2), $o_i(1 \leq i \leq k)$ should have been added to S prior to o. Therefore o could not pass the dominance check, which is in contradiction with the fact that o was added to S. Theorem is proved. ■

IV. Interaction Process

In this phase, our system will interact with end users in rounds. More specifically, the system at each round will choose a subset of objects from the candidate set generated in the previous phase and then present them to the users who will browse and pick one favourite object from them. The system keeps refining the user's preference that has been learnt based on the user's selections in current and all previous rounds. The interaction will continue until the user stops it explicitly or the system automatically decides to exit when it believes no more benefit of doing so. Theoretically the interaction can continue with sufficient rounds to test every pair of objects in the candidate set, such that we know exactly which k objects are preferred most by the users. However in practice this requires so large number of rounds (quadratic to the cardinality of candidate set) that no one is patient enough to go through this process. So the key problem of this phase is to strategically select a subset of candidate objects in each round such that this iterative process could converge quickly. Here we slightly abuse the notion of "converge" as in our problem it means getting a preference vector $\mathbf{w}'$ that can nicely approximate $\mathbf{w}$. Next we will first give a theoretical analysis to explain why we can get better approximation of user preference by involving user interaction. Then we describe the framework of the interaction process and propose several strategies for the subset selection from candidates.

Algorithm 1: Geo-textual Skyband Search (GSB)

```
Input: IR$^2$-Tree index of dataset: $tr$, query $q$, $k$
Output: skyband set $S$
1  Initialize an empty set $S$;
2  Initialize an empty min-heap $H$;
3  Add root node of $tr$ to $H$;
4  while $H$ is not empty do
5      $e \leftarrow$ top entry of $H$;
6      if $e$ is non-leaf entry then
7          for each child $e_i$ in $e$ do
8              if $e_i$ is dominated by less than $k$ tuples in $S$ then
9                  Add $e_i$ to $H$;
10     else
11         for each object $o_i$ in $e$ do
12             if $o_i$ is dominated by less than $k$ tuples in $S$ then
13                 Add $o_i$ to $S$;
14 return $S$;
```

A. Theoretical Analysis

Let S denote the candidate set generated from the previous phase. Given a spatial keyword query q with a location $q.\rho$ and m keywords $q.\psi : (t_1, t_2, ..., t_m)$, we can represent each object $o \in S$ with a $(m+1)$-length vector $\mathbf{o} : (1 - d(o.\rho, q.\rho), h_o(t_1), h_o(t_2), ..., h_o(t_m))$, where functions d and h are as defined in Definition 2. Then the utility function $u(o)$ (Eq. (1)) can be simply reformulated as $\mathbf{w}^{\intercal}\mathbf{o}$. Now suppose a subset $R \subset S$ has been selected and the user has picked an object o_i as her favourite within R. The choice of the user implies that the utility she can get from o_i is larger than any other object in R. Mathematically this can be represented by a set of inequalities on $u(o)$, i.e.,

$$u(o_i) > u(o_j), \forall o_j \in R \wedge o_j \neq o_i \quad (3)$$

After simple rewrite we get:

$$(\mathbf{o_i} - \mathbf{o_j})^{\intercal}\mathbf{w} > 0, \forall o_j \in R \wedge o_j \neq o_i \quad (4)$$

Since $\mathbf{w}$ is the unknown vector we would like to infer, the above inequalities can be treated as a set of linear constraints on $\mathbf{w}$. Let L^{α} denote the set of linear constraints obtained at the α-th round. Recall that we have another set of linear constraints for $\mathbf{w}$ in the first place, which is $0 < w_i < 1$, $\forall i \in [0, m+1]$. We denote them as L^0 as they are in place before the interaction starts. From geometrical perspective, L^{α} can be represented by the intersection of a set of half-spaces and L^0 corresponds to a hyper-cube with side-length

```
Algorithm 2: Interaction Framework
    Input: Candidate set $S$, query $q$
    Output: Linear constraint set $L$
 1  Initialize $L \leftarrow \{L^0\}$;
 2  $\alpha \leftarrow 1$;
 3  while true do
 4      $R \leftarrow$ SelectCandidate($S, q$);
 5      Present objects in $R$ to the user;
 6      $o_i \leftarrow$ user pick her favourite from $R$;
 7      $L^\alpha \leftarrow$ new linear constraints based on user's feedback;
 8      Add $L^\alpha$ into $L$;
 9      RefineCandidate($S, L$);
10      $\alpha \leftarrow \alpha + 1$;
11      if Terminate() is true then
12          Break;
13  return $L$;
```

of 1, which together form a convex polytope [13] in the $(m+1)$-dimensional space. Let L denote the union of all the linear constraints obtained up to the current round, i.e., $L = \{L^0, L^1, ..., L^\alpha\}$. As the interaction process goes on, more constraints will be added to L and the convex polytope formed by L will keep shrinking. In other words, the possible values that $\mathbf{w}$ can take are restricted within a space, which is getting smaller and smaller after receiving more user feedbacks. When the solution space to L is small enough, we can use any vector inside that space to approximate $\mathbf{w}$. The above analysis theoretically confirms that the goal and methodology of our work is feasible: the user preference vector $\mathbf{w}$ can be better estimated by involving user interaction.

B. Interaction Framework

The proposed interaction framework is illustrated by Algorithm 2, which will take the candidate set and query as input and return a set of linear constraints at the end of the iteration. The process starts with initializing L and α and then proceeds to a loop, in which a procedure *SelectCandidate()* will be invoked to select a subset R from S. The selection strategy taken by this procedure is pivotal to our framework and will be investigated more carefully later. After presenting all the objects in R and receiving user's favourite object, a new set of linear constraints L^α can be computed based on Eq. (4) and added into L. Then another important procedure *RefineCandidate()*, which will be discussed later as well, is invoked to make necessary changes to the candidate set by analysing the new constraint set L. At last, the termination condition of this interaction process is checked by the procedure *Terminate()*, which can be either manually specified by the user or automatically decided by the system. We are more interested in the latter case and will discuss mechanisms for automatic termination condition check in Section V.

C. Candidate Selection Strategy

SelectCandidate() is the most critical procedure in the interaction framework, which affects the quality of preference approximation and speed of convergence (i.e., the number of rounds needed before termination). Next we will discuss three selection strategies based on different heuristics.

1) Random Selection (RS) Strategy: The most straightforward approach to construct R is to randomly pick κ objects from S and present them to the user. The reason we want to select as many objects as possible (capped by κ) at each round is that the number of linear constraints we can get is proportional to the cardinality of R. The more constraints we have, the better estimation about $\mathbf{w}$ we are more likely to get. Still consider Figure 1 as an example and assume that we have got a candidate set $\{o_1, o_3, o_4, o_5, o_6\}$ (o_2 is not a candidate since it does not contain any query keyword). If $\kappa = 3$, we just randomly choose three objects from the candidate set, for instance o_1, o_3, o_4, and present them to the user.

2) Densest Subgraph (DS) Strategy: RS strategy suffers from the fact that it ignores the dominance relationship between candidates and thus may present object pairs, from which no effective constraint can be obtained. In particular, if there exist two objects o_i, o_j in R such that $o_i \prec o_j$, then the inequality $(\mathbf{o_i} - \mathbf{o_j})^\intercal \mathbf{w} > 0$ holds true for all $\mathbf{w}$. In other words, this inequality does not help estimate $\mathbf{w}$ better. Continuing with the previous example, if we present o_1, o_3, o_4 to the user, she will not pick o_3 as it is known to be dominated by o_1; if she picks o_1, the inequality derived from o_1, o_3 is also useless since we already know o_1 is better than o_3.

Essentially, in order to make most use of each round, R is most desirable if the expected number of constraints, denoted as $\mathcal{C}(R)$, that can be derived when the user pick any object from R is maximized. More formally,

$$R = \underset{R \subseteq S, |R| \leq \kappa}{\arg\max}\ E[\mathcal{C}(R)] = \underset{R \subseteq S, |R| \leq \kappa}{\arg\max} \sum_{o_i \in R} Pr[o_i]\mathcal{N}_R(o_i) \tag{5}$$

where $Pr[o_i]$ is the probability of o_i being picked by the user and $\mathcal{N}_R(o_i)$ is the number of objects in R that have no dominance relationship with o_i, i.e., the number of constraints that can be obtained if the user picks o_i. Let R' denote the subset of R, in which every object is not dominated by any object of R, i.e., $R' = \{o \in R | \nexists o' \in R, o' \prec o\}$. Assume a user has equal chance to choose any object in R', then we have

$$Pr[o_i] = \begin{cases} 1/|R'|, & \text{if } o_i \in R' \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

Taking Figure 1 as our example, if we choose o_1, o_3, o_4 as R, then $E[\mathcal{C}(R)] = 0.5 * 1 + 0.5 * 2 = 1.5$, since the user has equal chance to pick o_1 and o_4.

Theorem 2. *Finding the optimal R is an NP-complete problem.*

Proof: We prove by a reduction from k-clique problem, which is one of the 21 classic NPC problems [14]. We can construct an instance of graph $G(V, E)$ containing a κ-clique (a complete subgraph of size κ). Each vertex $v_i \in V$ corresponds to an object $o_i \in S$ and an edge $e = \langle v_i, v_j \rangle \in E$ indicates there is no dominance relationship between o_i and o_j. It is obvious that the optimal R corresponds to the κ-clique in G. If there exists a deterministic polynomial algorithm to find the optimal R, then the κ-clique can also be detected in polynomial time. ■

Since the optimal R is hard to find efficiently, we resort to heuristic methods that are simple, practical and efficient. We first define a notion of *dominance graph*.

Definition 8 (Dominance graph)**.** *Given a geo-textual dataset S, its dominance graph G_S is a graph wherein each vertex v_i represents an object $o_i \in S$ and each edge $\langle v_i, v_j \rangle$ indicates that no dominance relationship exists between v_i and v_j, i.e., $v_i \not\prec v_j$ and $v_j \not\prec v_i$.*

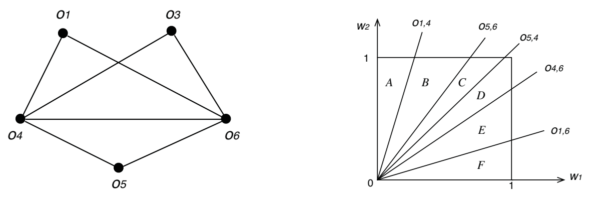
The dominance graph of the candidate set in Figure 1 is illustrated in Figure 2a. By modeling S as a *dominance graph* G_S, the problem of finding the optimal R is very similar with the problem of finding the densest subgraph, i.e., the subgraph with highest edge-to-vertex ratio (called density). Finding the densest subgraph with arbitrary size can solved in polynomial time[6] [16], [17] and linear algorithm exists for approximate solution [18]. The only difference is that in the densest subgraph problem all the vertices are considered when computing the density, while in our problem only the vertices not dominated by others are considered. Nevertheless, as we are not aiming at optimal solution, it suffices to find the (approximate) densest subgraph with arbitrary size first and adjust it in favour of Eq. (5). For example, by looking for the densest subgraph with three nodes in Figure 2a, we get an optimal R as $\{o_1, o_4, o_6\}$ ($\{o_3, o_4, o_6\}$, $\{o_4, o_5, o_6\}$ are also optimal). It is easy to validate that $E[\mathcal{C}(R)] = 2$ that is greater than $\{o_1, o_3, o_4\}$.

In the sequel, we only highlight some important steps of our method. Once the (approximate) densest subgraph has been found, we will be faced with one of the three possible situations: (1) $|R| > \kappa$; (2) $|R| = \kappa$; or (3) $|R| < \kappa$. The algorithm will step into a loop and take different actions for each case.

1) $|R| > \kappa$: we will remove the most dominant objects, the one that dominates the most other objects, in R until $|R| = \kappa$ and exit the loop;
2) $|R| = \kappa$: we will try to remove the most dominant object in R and test if $E[\mathcal{C}(R)]$ can be improved. If so, mark the removed object as visited and continues the loop; otherwise exit the loop;
3) $|R| < \kappa$: we will try to add one unvisited object from $S \setminus R$ that is adjacent to the most objects in R and test if $E[\mathcal{C}(R)]$ can be improved. If so, the process continues; otherwise exist the loop.

The rational behind our algorithm is to treat the densest subgraph as a good base and gradually adjust it (one object at a time) as long as the objective function can still be improved. The time complexity of this algorithm is linear since the number of loop does not depend on $|S|$. If we adopt the approximate algorithm [18] for the densest subgraph, which is also linear, the overall time complexity of the above procedure is linear.

3) Uncertainty Reduction (UR) Strategy: DS strategy aims to maximize the number of effective constraints that can be derived at each round, but does not differentiate the effectiveness of each constraint in estimating the preference vector. As mentioned earlier in this section, the set L^α of linear constraints obtained at round α together with the set L of all previous constraints form a convex polytope P and all possible values of $\mathbf{w}$ lie within P. According to information theory, the uncertainty of $\mathbf{w}$ can be modeled as its entropy, i.e.,



(a) Dominance graph (b) Geometric representation of constraints

Fig. 2: Selection strategies

$$h[\mathbf{w}] = -\int_{\mathbf{w} \in P} Pr[\mathbf{w}] \log Pr[\mathbf{w}] d\mathbf{w} \tag{7}$$

Assuming $\mathbf{w}$ has equal chance to take any value inside P, we have $h[\mathbf{w}] = \log V(P)$, where $V(P)$ is the volume of P. Therefore to reduce the uncertainty, we need to reduce the volume of P as much as possible. Based on this observation, we aim to find an R such that the expected volume of the derived polytope is minimized, formalized as the following objective function:

$$R = \underset{R \subseteq S, |R| \leq \kappa}{\arg\min}\ E[V(P)] = \underset{R \subseteq S, |R| \leq \kappa}{\arg\min} \sum_{o_i \in R} Pr[o_i] V(P_{o_i}) \tag{8}$$

where P_{o_i} is the polytope formed by the constraints based upon o_i being picked by the user, together with all previous constraints L. Figure 2b gives the geometric representations of some constraints about $\mathbf{w}$[7]. Each line labelled with $o_{i,j}$ represents a linear constraint derived from the pair $\langle o_i, o_j \rangle$. The squared area indicates the initial constraint before interaction. If we select $\{o_1, o_4, o_6\}$ as R and the user picks o_4, then the new uncertain space P_{o_4} will be reduced to the area $B \cup C \cup D$.

Greedy algorithm. Finding the optimal R according to Eq. (8) is also NP-complete (with similar proof of Theorem 2). To maintain the low latency of the interaction process, we propose a greedy algorithm based on a heuristic computed for each pair of candidate objects, which indicates its potential capability of reducing uncertainty of $\mathbf{w}$ when presented to the user.

For each pair of adjacent nodes o_i, o_j in the dominance graph G_S, the hyperplane $(\mathbf{o_i} - \mathbf{o_j})^\intercal \mathbf{w} = \mathbf{0}$ will divide the hypercube into two polytopes, denoted by $P^+(o_i, o_j)$ and $P^-(o_i, o_j)$. At the α-th round of interaction, let P^α denote the polytope formed by all the constraints obtained from previous rounds, i.e., L. Since $\mathbf{w}$ can take any value inside P^α with equal chance, when the pair of objects o_i, o_j is presented, a user's probability of choosing o_i is proportional to the relative volume of the intersection between $P^+(o_i, o_j)$ and P^α, i.e., $Pr[o_i] = V(P^+(o_i, o_j) \cap P^\alpha)/V(P^\alpha)$. Therefore the expected volume of the new polytope $P'(o_i, o_j)$ after the user's choice between o_i and o_j is

$$\begin{aligned} E[V(P'(o_i, o_j))] = {} & Pr[o_i] V(P^+(o_i, o_j) \cap P^\alpha) \\ & + (1 - Pr[o_i])(V(P^\alpha) - V(P^+(o_i, o_j) \cap P^\alpha)) \end{aligned} \tag{9}$$

[6]However, finding the densest subgraph with at most k vertices is proved to be NP-complete [15]

[7]This figure is for illustration purpose only. Actually they should be in three-dimensional space as their are two query keywords.

It is not difficult to see that Eq. (9) reaches its minimum value when $Pr[o_i] = 1/2$, which means the hyperplane $(\mathbf{o_i} - \mathbf{o_j})^\intercal \mathbf{w} = \mathbf{0}$ divides P^α into two halves. Hence our algorithm uses the volume ratio $\gamma(o_i, o_j)$ between the smaller newly intersected polytopes and P^α the heuristic, i.e.,

$$\gamma(o_i, o_j) = \frac{\min\{V(P^+(o_i, o_j) \cap P^\alpha), V(P^-(o_i, o_j) \cap P^\alpha)\}}{V(P^\alpha)} \tag{10}$$

Higher $\gamma(o_i, o_j)$ implies more uncertainty could be reduced if the user chooses any one between o_i and o_j. Therefore our greedy algorithm will add one pair of objects $o_i, o_j \in S$ into R at a time in descending order of $\gamma(o_i, o_j)$ until $|R| = \kappa$. Easy to see the time complexity of this algorithm is quadratic to the number of candidates since it needs to calculate γ for each pair of objects. Consider all the constraints in Figure 2b. If P^α refers to the original uncertain space (i.e., the unit square), then the pair $\langle o_4, o_5\rangle$ will be chosen first since the line $o_{4,5}$ almost equally divides the square. Nevertheless if P^α comes down to $B \cup C \cup D$, $\langle o_5, o_6\rangle$ becomes the most promising pair.

Efficient approximation. However computing the exact volume of a polytope in multi-dimensional space is an expensive process in general case [19], so we resort to efficient approximation methods. The basic idea of our approach is to approximate a polytope with a finite set of points and translate the problem of calculating volume of polytope to joint set counting problem, which can be solved more efficiently. We first randomly generate a set of points $X : \{x_1, x_2, ..., x_M\}$ in the $(m+1)$-dimensional unit hypercube, that is $X \subset [0, 1]^{m+1}$. Then the volume of each polytope can be approximated by the cardinality of a subset of X as follows:

$$\begin{aligned} V(P^\alpha) &\approx |X^\alpha : \{x \in X | x \text{ satisfies all constraints in } L\}| \\ V(P^+(o_i, o_j)) &\approx |X^+(o_i, o_j) : \{x \in X | (o_i - o_j)x > 0\}| \\ V(P^-(o_i, o_j)) &\approx |X^-(o_i, o_j) : \{x \in X | (o_i - o_j)x < 0\}| \end{aligned} \tag{11}$$

Here a better approximation of the polytopes can be obtained by increasing M, though it would incur more computational cost. Since all the polytopes have been approximated by subsets of $\mathbf{X}$, we can approximate $\gamma(o_i, o_j)$ as follows:

$$\gamma(o_i, o_j) \approx \frac{\min\{|X^+(o_i, o_j) \cap X^\alpha|, |X^-(o_i, o_j) \cap X^\alpha|\}}{|X^\alpha|} \tag{12}$$

However it still requires iterating through all pairs of candidates and calculating their γ. To further improve the efficiency, we index the set $X^{+/-}(o_i, o_j)$ for each pair of adjacent nodes $\langle o_i, o_j\rangle$ in G_S with an inverted list at the beginning of interaction phase. Note that we only need to index the one between $X^+(o_i, o_j)$ and $X^-(o_i, o_j)$ with fewer points to save index space. Afterwards, given an X^α at each round, we look up each entry $x \in X^\alpha$ in the inverted list and in the meantime maintain a counter $C(o_i, o_j)$ for each encountered pair in the list. At last the value $C(o_i, o_j)$ is exactly the cardinality of the joint set $X^+(o_i, o_j) \cap X^\alpha$ (or $X^-(o_i, o_j) \cap X^\alpha$). Now we only need to sort all the encountered pairs in their ascending order of $|C(o_i, o_j) - |X^\alpha|/2|$ and pick them from the beginning of the ranked list. Figure 3 illustrates the above process using our running example. The grey areas represent the subsets $X^+(o_5, o_4)$ and $X^-(o_4, o_6)$ respectively and the coloured area represents X^α. Red shaded area in the inverted list means the entries that overlap with X^α and thus get accessed. At the end the pair $\langle o_5, o_4\rangle$ is ranked the highest. We can see that indeed the line $o_{5,4}$ almost divides the coloured area equally, which means the approximation is quite effective.

Summary: The essential difference between DS and UR strategies lies in their different objectives. DS tries to maximize the expected number of effective constraints that can be obtained at each round, while UR aims to reduce the expected volume of uncertain space of $\mathbf{w}$ as much as possible. To some extent, we can regard DS as a quantity-oriented strategy and UR as a quality-oriented strategy.

D. Candidate Refinement

As outlined by the interaction framework, after a new constraint set L^α has been added to L, a function *RefineCandidate()* will be invoked. Its main purpose is to reduce the number of candidates that need to be considered in the next round of interaction by taking the following actions on the dominance graph G_S:

- For each adjacent pair of nodes o_i, o_j in G_S, if it can be inferred from L^α that o_i is superior/inferior than o_j, then the edge between o_i and o_j is removed;
- For each object $o_i \in S$, if there exist more than $k-1$ objects in S that are superior, either by dominance or inference, than o_i, then o_i is removed from S.

V. Finalisation

In this section we briefly discuss the finalisation phase of the interactive query processing, which includes the explanation of the function *Terminate()* and final estimation of $\mathbf{w}$.

A. Termination condition

In Algorithm 2, the subroutine *Terminate()* checks whether the interaction phase should terminate. This can be explicitly instructed by the user when she does not want to continue the interaction, but more preferably it should be done by the system automatically. Recall that the uncertainty of $\mathbf{w}$ can be measured by the volume of the polytope P^α at each round α, which keeps decreasing as the interaction goes on. At the end of each round, we can examine if the ratio $\frac{V(P^\alpha)}{V(P)}$ is below a certain termination threshold $\tau \in (0, 1)$. Setting a high τ means the interaction is easier to terminate (converge) but results in a more uncertain preference; and vice versa.

B. Estimation of $\mathbf{w}$

Once the interaction phase has been terminated, it will output a constraint set L. Any vector $\mathbf{w}'$ subject to L can be a valid estimation of $\mathbf{w}$. Inspired by the theory behind SVM techniques, we try to find the $\mathbf{w}'$ with the highest confidence to separate superior and inferior objects. This translates to minimizing $||\mathbf{w}'||^2$ subject to all the constraints in L, where $||\cdot||$ is the 2-norm of a vector. The optimal value of $\mathbf{w}'$ can be obtained by any Linear Programming solver such as LpSolve[8] or many mathematical libraries[9]. Finally, we issue

[8]http://sourceforge.net/projects/lpsolve/

[9]For instance, Apache Commons Mathematics Library, http://commons.apache.org/proper/commons-math/

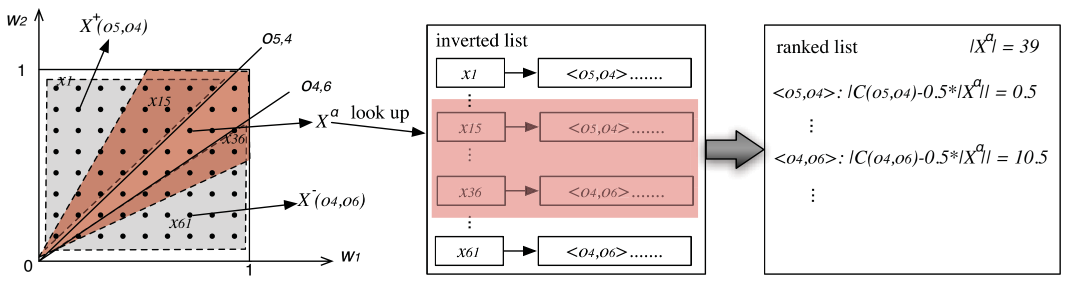


Fig. 3: Efficient approximation of UR strategy

TABLE III: Statistics of dataset

Attribute	CH	NY
Total number of PoIs	8,203,485	206,416
Total number of unique keywords	154,904	87,394
Average number of unique keywords	8	18

a top-k query with $\mathbf{w}'$ to retrieve the top$k$ results from the remaining candidates.

VI. Experimental Evaluation

A. Experimental Settings

Algorithms. We study the performance of the GSB algorithm in Sect. III for retrieving candidates, and the three strategies for selecting candidates, namely RS, DS and UR, proposed in Sect. IV. All algorithms were implemented in Java on Windows 7, and run on an Intel(R) CPU i7-3770 @3.40GHz with 16GB RAM.

Data and queries. We use two real PoI datasets crawled from online SNS for our experimental study. The first dataset (CH) is crawled from Sina Weibo[10] and contains around 8 million PoIs in China. Each PoI has a name, location (in the form of longitude, latitude) and category tags (with several subcategories). We combine the name and categories as the textual information of each PoI. The second dataset (NY) contains around 200 thousand PoIs in New York referred by the check-in records of Foursquare[11]. We use the name, tags and check-in comments as the textual information for each PoI. The detailed statistics of the datasets are summarized in Table III. These two datasets are quite different in terms of the size, area of distribution and the number of keywords per object.

To generate a spatial keyword query, we first randomly pick an object in the dataset and regard its location as the query location. Then we randomly choose a specified number of words from the object as the query keywords. This object is temporarily excluded from the database during the query execution. Each query set contains 100 queries and the average performance is reported.

B. Experimental Results on CH Dataset

All the parameters and their default values in the experiments on CH dataset are summarized in Table IV.

[10]http://www.weibo.com
[11]http://foursqure.com

TABLE IV: Experiment parameter settings

Parameter	Values (default in bold)
Data cardinality	0.5M, 1M, 2M, 4M, **8M**
Number of query keywords	2,3, **4**, 5, 6
k	5, 10, **20**, 50, 100
Number of presented candidates κ	2, 4, **6**, 8, 10
Termination threshold τ	0.2, 0.4, 0.6, 0.8
Number of rounds	3

1) Experiments on Candidate Generation Phase: We first test the performance of GSB algorithm proposed in Sect. III. Since the performances of IR^2-tree with different settings have been studied extensively in [2], we only report our experimental results of GSB algorithm on IR^2-tree with 4K page size, 1GB LRU buffer size and signature length of 7000. Each set of experiments measure the CPU time and the number of I/Os. The CPU time excludes the time cost for loading data from disk, which dominates the computational time cost. Rather than measuring physical I/Os, we report simulated I/O costs. If a leaf node is visited, the cost is increased by 1, and if a signature file is loaded, the cost is increased by the number of disk pages used for storing the signature file.

Baseline. Since there is no existing algorithm for geo-textual skyline search, we devise a baseline method as follows. Each object is indexed by R-tree based on its spatial location and inverted index based on its keywords respectively. The baseline algorithms will first look up the inverted lists to obtain the objects containing at least one query keyword, and then apply the BNL skyline search algorithm ([10] etc.) to get the final results. The simulated I/O costs are measured as the number of accessed page blocks for storing the inverted file.

Effect of data cardinality. To evaluate the scalability of the GSB algorithm, four additional datasets are generated by sampling CH from 0.5 million to 4 million PoIs. As shown in Figure 4a, while the CPU time and I/O costs of both methods increase with the cardinality of the dataset, the scalability of GSB algorithm is much better. We find that when the dataset is relatively small (0.5M), the I/O cost of GSB is even greater than that of the baseline method. A possible explanation is, the I/O cost of baseline method is only dependent on the number of objects sharing common keywords with the query, while GSB algorithm may need to access extra nodes that are farther away from query location when data become sparser.

Effect of query keywords. Figure 4b shows the performances of both algorithms when the number of query keywords vary from 2 to 6. We did not further increase this number as in

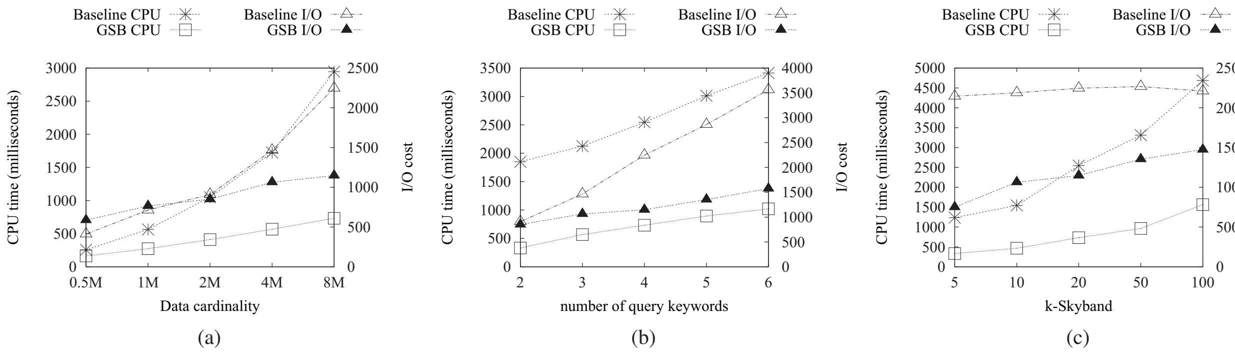


Fig. 4: Performance evaluation of GSB algorithm (CH)

practice a user would not bother to type too many keywords. As expected, increasing the number of query keywords will incur more CPU time and I/O costs for both algorithms. This is because (1) it is getting harder for an object to be dominated when more keywords are involved in determining the dominance relation and thus more objects become skyband; and (2) the chance of a node being filtered using the signature is lower; and (3) more inverted lists in the baseline algorithm need to be retrieved when there are more query keywords.

Effect of k. We also evaluate the performance of candidate generation by varying k from 5 to 100. From Figure 4c we observe that the I/O cost of the baseline method is not affected but its CPU time increases dramatically. This is due to the fact that the same amount of objects are loaded from disk regardless of how k changes. Nevertheless a greater k means more dominance checks need to be performed against all the candidates. The CPU time and I/O cost of GSB algorithm also rise with k but at much lower rates.

2) Experiments on Interaction Phase: Next we evaluate the performance of different candidate selection strategies proposed in Sect. IV. In particular, we evaluate both effectiveness and efficiency of the three strategies, namely Random Selection (RS), Densest Subgraph (DS) and Uncertainty Reduction (UR). For the UR method, we generate 10K points to uniformly distribute in the preference space. We perform three rounds of interactions for each query. The efficiency is measured as the average running time per round during interaction.

Effectiveness measure. To measure the effectiveness, for each query we randomly generate a vector $\mathbf{w}$ to simulate a user's preference. At each round, the candidate with the highest utility score w.r.t. $\mathbf{w}$ is picked and finally $\mathbf{w}'$ is estimated. Let π and π' denote the top-k results based on $\mathbf{w}$ and $\mathbf{w}'$ respectively. We adopt a highly cited distance function proposed by Fagin et al [20] to compare two top-k lists, i.e.,

$$F(\pi,\pi') = \sum_{o\in\pi\cap\pi'} |\pi(o)-\pi'(o)| + 2(k-|\pi\cap\pi'|)(k+1) - \sum_{o\in\pi\setminus\pi'} \pi(o) - \sum_{o\in\pi'\setminus\pi} \pi'(o) \tag{13}$$

where $\pi(o)$ represents the position of object o in the top-k list π. It is easy to prove the inequality $0 \le F(\pi,\pi') \le k(k+1)$ holds. F reaches the minimum when two lists are exactly the same and maximum when they are completely different. We translate this distance into an accuracy measure as follows.

$$Accuracy = 1 - \frac{F(\pi,\pi')}{k(k+1)} \tag{14}$$

In each effectiveness test, we also construct a baseline top-k list using an equal-weighted preference vector and compare it with the interactive methods. Next we present the effect of k, the number of query keywords and the number of objects displayed to users at each round. We do not show the effect of the database cardinality since it does not affect the interaction performance once all the candidates have been retrieved.

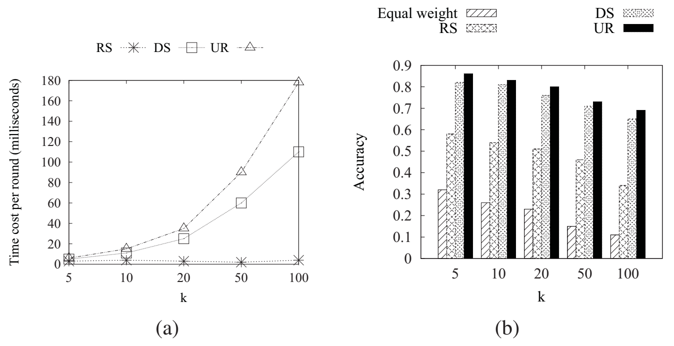


Fig. 5: Interaction performance with varying k (CH)

Effect of k. As demonstrated in Figure 5a, DS and UR become more time consuming when k increases since more candidates are generated. The running time of UR climbs up more quickly than DS because it examines (almost) every pair of candidates. Nor surprisingly, random selection is the most efficient strategy and consumes constant time regardless of k. Regarding their accuracies (Figure 5b), UR performs the best followed by DS closely and then RS. It is interesting to see that even RS can outperform the equal-weighted vector by a large margin. All accuracies deteriorate with greater k since it is more likely to include incorrect objects within a longer top-k list.

Effect of query keywords. The number of query keywords has similar effect on both efficiency and accuracy of interaction phase, as demonstrated in Figure 6a and 6b. This is because the number of query keywords means the dimensionality of

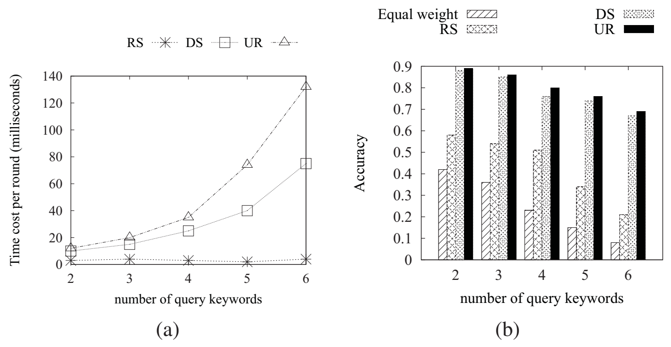


Fig. 6: Interaction performance with varying query keywords (CH)

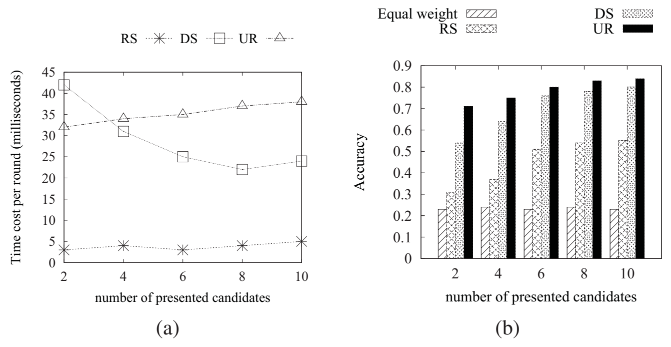


Fig. 7: Interaction performance with varying κ (CH)

the user's preference, thus the more query keywords, the more candidates can be generated and the higher uncertainty there is in the user's preference.

Effect of κ. Figure 7a shows the time costs of different selection strategies by varying number of candidates presented at each round from 2 to 10. We do not further increase κ since it is difficult to identify the favourite object from too many candidates in practice. Besides of RS, which remains unaffected, UR is also insensitive to this parameter due to the fact that the predominant cost of UR lies in ranking all pairs of candidates according to Eq. (12) that is independent of κ. An interesting observation is that DS becomes less efficient when presenting fewer candidates. To explain this, recall that DS will first find the densest subgraph and adjust it based on the relationship between the subgraph cardinality and κ. Usually the densest subgraph cardinality is greater than the chosen κ, so it takes extra time to drop the vertices with the least edges. Figure 7b demonstrates that all the interaction based methods lead to higher accuracies when presenting more candidates per round. This is as expected since more constraints on the preference vector are potentially to be derived. We also notice that the marginal benefit of increasing κ is more obvious for DS than UR because DS needs more candidates presented in order to obtain more constraints while UR relies more on the power of individual constraint in terms of the ability to reduce uncertainty.

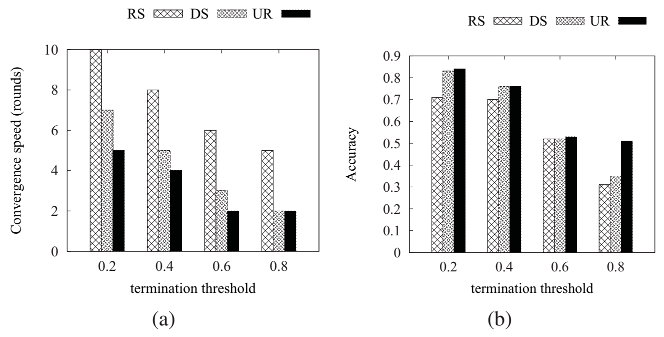


Fig. 8: Effect of τ (CH)

3) Experiments on Finalisation Phase: The last set of experiments evaluate the effect of τ, the termination threshold in Sect. V, on the query accuracies and convergence speeds of the three selection strategies. The convergence speed is measured as the number of rounds needed before the termination condition is satisfied. We cap the maximum number of rounds to be 10 to avoid the cases when τ is hard to achieve. Figure 8a compares the convergence speeds of three selection strategies. Generally RS has much slower convergence speed than the other two methods, especially it cannot even terminate automatically when τ is set too low. UR can converge more quickly than DS especially for low τ as it aims to reduce the volume of the preference space as much as possible at each round. Figure 8b shows that all accuracies decline as the threshold is raised (relaxed). Moreover, once τ is set, the accuracies of all strategies are quite similar since the uncertainty in the final preference is decided by τ.

C. Experimental Results on NY Dataset

We test the same set of parameters on NY dataset and observe the similar results. The main difference with CH dataset is that: (1) time and I/O costs of GSB algorithm are lower because NY is a smaller dataset and the distribution of objects is more concentrated; (2) interaction phase becomes more efficient since less candidates have been generated. An explanation is that the average number of unique keywords on each object in NY is greater than CH, which means there is higher chance for fewer objects to dominate the rest. Due to space limitation, we show the complete experiment results on NY in our technical report (http://staff.itee.uq.edu.au/kevinz/papers/itksk2014.pdf).

VII. Related Work

Spatial keyword queries. Searching geo-textual objects with query location and keywords has gained increasing attention recently due to the popularity of location-based services. Besides of the work that have been classified in Sect. I, many variants of spatial keyword queries have also been proposed such as moving spatial keyword query [21], reverse spatial-textual nearest neighbor query [22], m-closest keyword queries [23][24], approximate spatial keyword query [25], direction-aware spatial keyword query [26], region based spatio-textual query [27] and so on. While various novel indexes and query processing algorithms have been developed, they all assume the users know exactly about their preferences between spatial proximity and textual relevance, so query efficiency is the only focus of these work. On the other hand, we observe this assumption may not be always realistic and thus investigate how to automatically learn the user's preference through interactions. Therefore we focus on both the effectiveness of learning and efficiency of interaction process.

Interactive queries. There are also papers that consider inferencing the utility function by requiring feedbacks from the user. Mindolin et al. [28] propose the p-skyline query which is a framework that assumes that different attributes have varying levels of importance. This enables the system to rank the tuples and control the output size. To avoid asking users directly for weights, they offer an alternative approach to discover importance from user feedback, i.e., the user has to partition example tuples to desirable and undesirable groups. Jiang et al. [29] propose to mine user preferences on some categorical attributes of tuples that have no natural partial order. Their interaction process requires the user to pick superior and inferior examples from the presented tuples. In our paper, we ask users to pick one tuple per round, which is less demanding than the partition-based feedback approach but more challenging since less information can be obtained each round. We adopt the same interaction style with the work [9]. However, as discussed in Sect. I, it *constructs* virtual tuples rather than *searching* existing tuples to present at each round, which means their method cannot be applied to solve our problem in this work.

VIII. Conclusion

In this work we have analysed the hardness of specifying preference weights between spatial proximity and keyword relevance in conventional top-k spatial keyword queries and proposed the ITkSK query that can learn the users' preferences automatically based on their feedback. Our solution starts with finding the k geo-textual skyband objects as the initial candidates, then strategically presents a subset of candidates to the user at each round during the interaction iteration and finally retrieve the top-k tuples based on the estimated preference vector. Extensive experiments based on real PoI datasets have been conducted and the favourable results have confirmed that the quality of top-k spatial keyword queries can be enhanced significantly with even a small number of rounds. In future it is also interesting to investigate the effect of other interaction approach, e.g., allowing the user to pick multiple favourable objects per round.

Acknowledgement

This work was partially supported by Australian Research Council DP120102829, ARC DECRA DE140100215 and Natural Science Foundation of China (Grant No.61232006).

References

[1] X. Cao, L. Chen, G. Cong, C. S. Jensen, Q. Qu, A. Skovsgaard, D. Wu, and M. L. Yiu, "Spatial keyword querying," in *Conceptual Modeling*. Springer, 2012, pp. 16–29.

[2] L. Chen, G. Cong, C. S. Jensen, and D. Wu, "Spatial keyword query processing: an experimental evaluation," in *Proceedings of the 39th international conference on Very Large Data Bases*. VLDB Endowment, 2013, pp. 217–228.

[3] R. Hariharan, B. Hore, C. Li, and S. Mehrotra, "Processing spatial-keyword (sk) queries in geographic information retrieval (gir) systems," in *SSBDM*. IEEE, 2007, pp. 16–25.

[4] A. Cary, O. Wolfson, and N. Rishe, "Efficient and scalable method for processing top-k spatial boolean queries," in *Scientific and Statistical Database Management*. Springer, 2010, pp. 87–95.

[5] I. De Felipe, V. Hristidis, and N. Rishe, "Keyword search on spatial databases," in *ICDE*. IEEE, 2008, pp. 656–665.

[6] X. Cao, G. Cong, and C. S. Jensen, "Retrieving top-k prestige-based relevant spatial web objects," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 373–384, 2010.

[7] G. Cong, C. S. Jensen, and D. Wu, "Efficient retrieval of the top-k most relevant spatial web objects," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 337–348, 2009.

[8] J. Zobel and A. Moffat, "Inverted files for text search engines," *ACM Computing Surveys*, vol. 38, no. 2, pp. 1–56, 2006.

[9] D. Nanongkai, A. Lall, A. Das Sarma, and K. Makino, "Interactive regret minimization," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. ACM, 2012, pp. 109–120.

[10] S. Borzsonyi, D. Kossmann, and K. Stocker, "The skyline operator," in *ICDE*, 2001, pp. 421–430.

[11] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "Progressive skyline computation in database systems," *ACM Transactions on Database Systems (TODS)*, vol. 30, no. 1, pp. 41–82, 2005.

[12] N. Roussopoulos, S. Kelley, and F. Vincent, "Nearest neighbor queries," in *SIGMOD*, 1995, pp. 71–79.

[13] B. Grunbaum, V. Klee, M. A. Perles, and G. C. Shephard, *Convex polytopes*. Springer, 1967.

[14] R. M. Karp, *Reducibility among combinatorial problems*. Springer, 1972.

[15] R. Andersen and K. Chellapilla, "Finding dense subgraphs with size bounds," in *Algorithms and Models for the Web-Graph*. Springer, 2009, pp. 25–37.

[16] G. Gallo, M. D. Grigoriadis, and R. E. Tarjan, "A fast parametric maximum flow algorithm and applications," *SIAM Journal on Computing*, vol. 18, no. 1, pp. 30–55, 1989.

[17] A. V. Goldberg, *Finding a maximum density subgraph*. University of California Berkeley, CA, 1984.

[18] G. Kortsarz and D. Peleg, "Generating sparse 2-spanners," *Journal of Algorithms*, vol. 17, no. 2, pp. 222–236, 1994.

[19] M. E. Dyer and A. M. Frieze, "On the complexity of computing the volume of a polyhedron," *SIAM Journal on Computing*, vol. 17, no. 5, pp. 967–974, 1988.

[20] R. Fagin, R. Kumar, and D. Sivakumar, "Comparing top k lists," *SIAM Journal on Discrete Mathematics*, vol. 17, no. 1, pp. 134–160, 2003.

[21] D. Wu, M. Yiu, C. Jensen, and G. Cong, "Efficient continuously moving top-k spatial keyword query processing," in *ICDE*, 2011.

[22] J. Lu, Y. Lu, and G. Cong, "Reverse spatial and textual k nearest neighbor search," in *SIGMOD*, 2011.

[23] D. Zhang, Y. Chee, A. Mondal, A. Tung, and M. Kitsuregawa, "Keyword search in spatial databases: Towards searching by document," in *ICDE*, 2009, pp. 688–699.

[24] D. Zhang, B. Ooi, and A. Tung, "Locating mapped resources in web 2.0," in *ICDE*, 2010, pp. 521–532.

[25] B. Yao, F. Li, M. Hadjieleftheriou, and K. Hou, "Approximate string search in spatial databases," in *ICDE*, 2010, pp. 545–556.

[26] G. Li, J. Feng, and J. Xu, "Desks: Direction-aware spatial keyword search," in *ICDE*, 2012.

[27] X. Cao, G. Cong, C. S. Jensen, and M. L. Yiu, "Retrieving regions of interest for user exploration," *Proceedings of the VLDB Endowment*, vol. 7, no. 9, 2014.

[28] D. Mindolin and J. Chomicki, "Discovering relative importance of skyline attributes," *PVLDB*, vol. 2, no. 1, pp. 610–621, 2009.

[29] B. Jiang, J. Pei, X. Lin, D. W. Cheung, and J. Han, "Mining preferences from superior and inferior examples," in *SIGKDD*. ACM, 2008, pp. 390–398.