Temporal Spatial-Keyword Top-k Publish/Subscribe

Lisi Chen #1, Gao Cong #2, Xin Cao †3, Kian-Lee Tan *4

Nanyang Technological University, {1lchen012@e., 2gaocong@}ntu.edu.sg
† *Queen's University Belfast,* 3x.cao@qub.ac.uk
* *National University of Singapore,* 4tankl@comp.nus.edu.sg

***Abstract*—Massive amount of data that are geo-tagged and associated with text information are being generated at an unprecedented scale. These geo-textual data cover a wide range of topics. Users are interested in receiving up-to-date tweets such that their locations are close to a user specified location and their texts are interesting to users. For example, a user may want to be updated with tweets near her home on the topic "food poisoning vomiting." We consider the Temporal Spatial-Keyword Top-k Subscription (TaSK) query. Given a TaSK query, we continuously maintain up-to-date top-k most relevant results over a stream of geo-textual objects (e.g., geo-tagged Tweets) for the query. The TaSK query takes into account text relevance, spatial proximity, and recency of geo-textual objects in evaluating its relevance with a geo-textual object. We propose a novel solution to efficiently process a large number of TaSK queries over a stream of geo-textual objects. We evaluate the efficiency of our approach on two real-world datasets and the experimental results show that our solution is able to achieve a reduction of the processing time by 70–80% compared with two baselines.**

I. Introduction

Massive amount of data that contain both text information and geographical location information are being generated at an unprecedented scale on the Web. For example, Tweets, each containing up to 140 characters, can be associated with locations, which may be coordinates (latitude and longitude) or semantic locations; and some social photo sharing websites (e.g., Flickr) contain photos with both descriptive tags and geographical information. As another example, check-ins or reviews in location based social networks (e.g., Foursquare) contain both text descriptions and locations of points of interest (POIs). We refer to such data with both textual content and geographical content as *geo-textual objects*.

These geo-textual objects can be modeled as continuously arriving streams, and many real-world applications can benefit from a publish/subscribe system for streams of such geo-textual objects. The first example is annotation of POIs. Social updates (e.g., Tweets) often offer the quickest first-hand reports of news events [24], comments and reviews, indicating the public view, business promotion information, etc. In order to provide users a better service, a POI service provider (e.g., Yelp) may want to annotate each POI with its up-to-date relevant tweets in terms of both text relevance and spatial proximity. In addition, a manager for a POI may be interested in up-to-date tweets whose locations are close to each POI and whose text is relevant to the description of the POI. The second example application is that Groupon customers register their locations and keywords of their interests, and Groupon pushes to customers the Groupon messages whose locations are close to customers' locations and whose text is relevant to their interest keywords. As another example, users on Twitter want to be updated with tweets near their home on a topic (e.g., food poisoning vomiting). In these applications, user would prefer to be updated with a few most relevant tweets in terms of distance, text relevance, and recency, rather than being overwhelmed by a large number of tweets.

The existing publish/subscribe systems for streams of geo-textual objects [4], [13], [26] allow users to receive up-to-date geo-textual objects whose locations have spatial overlap with a user specified region and texts contain the user specified keywords. In these systems [4], [13], [26], both the user specified spatial region and the user specified keywords in a subscription query perform as boolean filters. Such subscription query has two problems: (1) A subscriber may receive very few matching geo-textual objects or may be overwhelmed by a huge volume of matching objects, depending on the specified query region and query keywords; (2) It may be difficult for a subscriber to specify the query keywords, and especially the size of a spatial region when they are used as boolean filters. For example, a larger region may result in too many results while a small one may result in no result.

To address these issues, in this work we consider the top-k subscription query, for which we rank-order geo-textual objects and return only the top-ranked objects. The rationale behind is analogous to the reason that search engines rank-order documents matching a query rather than employ the boolean retrieval model (e.g., the resulting number of matching documents of a boolean filter can far exceed the number a human user could possibly sift through [19]). The recently proposed publish/subscribe system for tweets [24] returns a subscriber top-k tweets that are ranked based on both keyword relevance and recency. However, it does not consider the spatial information.

In this paper, we propose a new type of top-k subscription query, which is referred to as Temporal Spatial-Keyword Top-k Subscription (TaSK) query, where TaSK queries are treated as subscriptions and geo-textual objects are published items in the publish/subscribe system. The TaSK query takes into account the following three aspects in evaluating its relevance with a geo-textual object: (1) Text relevance; (2) Spatial proximity; (3) Recency of geo-textual object. The TaSK query continuously maintains its up-to-date top-k results over a stream of geo-textual objects (e.g., tweets with locations). A TaSK query is triggered by a new published item (e.g., a tweet with location) only if the new tweet scores higher than the current k-th top result tweet.

Challenges: We aim at maintaining the up-to-date top-k results for a large number of TaSK queries, which arrive as a stream, over a stream of geo-textual objects. A straightforward

approach would work as follows: for each new geo-textual object o we compute its ranking score w.r.t. each TaSK query q; If the score is larger than the ranking score of the current kth result of q, o becomes a result and is used to update the current top-k results for q. Note that the ranking score of each object in the top-k results of q declines over time and we need to recompute them each time when a new object arrives. The straightforward approach is computationally expensive when the number of queries is large or the geo-textual objects arrive at a high rate. Hence, we need a more efficient mechanism to handle TaSK queries over geo-textual data stream.

We find that an underlying idea of many publish/subscribe systems and continuous query processing systems (e.g., [1], [24], [27]) is to group similar subscription queries such that they can be evaluated simultaneously for a new published object (more discussion can be found in Section III), thus improving the performance of query processing. However, it is challenging to achieve the similar optimization for TaSK queries. In this paper, we propose an approach including the following key techniques to representing, grouping, and indexing TaSK queries such that each group of queries can be processed simultaneously for a new geo-textual object.

(1) We propose a new concept *Conditional Influence Region* (CIR) to represent the TaSK query, and utilize CIR to design a filtering condition of a TaSK query w.r.t. a spatial cell to determine whether a new object is a result of the query. Based on this, we develop an approach to grouping and indexing TaSK queries and generating filtering conditions for each group of queries such that they can be evaluated simultaneously. (Section IV)

(2) We develop an algorithm for making use of the filtering conditions (of each group of queries on each spatial cell) to efficiently retrieve the TaSK queries that have a new geo-textual object as one of their top-k results. (Section V)

(3) In our method, each TaSK query needs to be associated with a set of non-overlapping spatial cells that can cover the whole spatial area. To select a set of cells for better performance, we propose a cost model based approach. (Section VI)

In summary, the paper's contributions are twofold. First, we define the TaSK query and present the first study on the problem of maintaining the up-to-date results for a large number of TaSK queries over a stream of geo-textual objects. In particular, we propose a novel approach comprising the aforementioned key techniques. Second, we conduct an extensive experimental study for evaluating the paper's proposals on real-world datasets of a large scale, collected from FourSquare and Twitter. The experimental results suggest that our proposed algorithm is able to achieve a reduction of the processing time by 70% to 80% compared with two baselines developed based on existing techniques.

II. Problem Statement

We define the geo-textual object and the Temporal Spatial-Keyword Top-k Subscription (TaSK) query.

Definition 1: Geo-Textual Object. A geo-textual object is represented with a triple $o = \langle \psi, \rho, t_c \rangle$, where ψ is a set of keywords, ρ is a location point with latitude and longitude, and t_c is the creation time of object o. □

In this paper, we consider a stream of geo-textual object data. For example, it can be geo-tagged tweets in Twitter, geo-tagged photos with tags in Flickr, check-ins with text descriptions in Foursquare, geo-tagged webpages, etc.

Intuitively, given a stream of geo-textual objects, a TaSK query is to continuously retrieve k objects over time such that these objects are relevant to the query keywords, their locations are close to the query location, and they are fresh. Note that in addition to text relevance and spatial proximity, recency is important for geo-textual data streams. For example, tweets are often tied to some event and their relevance to a query declines as time passes [24].

Definition 2: Temporal Spatial-Keyword Top-k Subscription (TaSK) Query. A TaSK query $q = \langle \psi, \rho, k, \alpha \rangle$, where ψ is a set of query keywords, ρ is the query location, k is the number of results to be maintained, and $\alpha \in [0, 1]$ is a preference parameter that balances the importance between distance proximity and text relevance, aims to continuously feed the user with new geo-textual objects whose temporal spatial-keyword scores are ranked within the top-k. The temporal spatial-keyword score of a geo-textual object o at time t_e is defined as follows.

$$S_{tsk}(q, o, t_e) = S_{sk}(q, o) \cdot S_t(o.t_c, t_e), \quad (1)$$

where $S_{sk}(q, o)$ computes the spatial-keyword relevance between query q and object o and $S_t(o.t_c, t_e)$ computes the object recency. Following previous work (e.g., [5], [6], [15], [22], [30]) we compute the spatial-keyword relevance $S_{sk}(q, o)$ between q and o as follows.

$$S_{sk}(q, o) = \alpha \cdot S_p(dist(q.\rho, o.\rho)) + (1 - \alpha) \cdot TRel(q.\psi, o.\psi), \quad (2)$$

where $S_p(dist(q.\rho, o.\rho))$ is the *spatial proximity score* of the distance between query q and object o, $TRel(q.\psi, o.\psi)$ indicates the *text relevance* between q and o. □

Example 1: Consider the annotation application in Section I. Each POI (e.g., coffee shop) will correspond to a TaSK query, where the location of the POI is the query location $q.\rho$ and its text description (e.g., coffee, espresso, mocha, green tea latte) corresponds to the query keywords $q.\psi$, and we want to continuously feed each POI with the top-k tweets with the highest temporal spatial-keyword scores. □

Note that all the textual, spatial, and temporal information are important in continuously retrieving top-k results. If we ignore the spatial (or temporal) aspect in the ranking function, even if users can get tweets that are relevant to the query keywords, their distances may be faraway from the query location (or they are not the most recent).

The spatial proximity score is calculated by the normalized Euclidian distance: $S_p(dist(q.\rho, o.\rho)) = 1\text{-}\frac{dist(q.\rho, o.\rho)}{dist_{max}}$,where $dist(q.\rho, o.\rho)$ is the Euclidian distance between q and o, and $dist_{max}$ can be the maximal possible distance in the spatial area. The text relevance can be computed using any information retrieval model in our method. We use language models [6] in this work, described as follows:

$$TRel(q.\psi, o.\psi) = \prod_{w \in q.\psi} PS(o.\psi, w), \quad (3)$$

where $PS(o.\psi, w)$ is the partial score of text relevance for $o.\psi$ w.r.t. keyword w, which is computed as follows:

$$PS(o.\psi, w) = (1-\lambda)\frac{Num(o.\psi, w)}{|o.\psi|} + \lambda\frac{Num(Coll, w)}{|Coll|},$$

where $Num(o.\psi, w)$ indicates the count of w in $o.\psi$, $Num(Coll, w)$ represents the count of w in the object collection $Coll$, and λ is a smoothing parameter of the Jelinek-Mercer smoothing method.

The recency of object o is calculated by the following exponential decay function:

$$S_t(o.t_c, t_e) = D^{-(t_e - o.t_c)}, \quad (4)$$

where D is base number that determines the rate of the recency decay. The function is monotonically decreasing with $t_e - o.t_c$. It is introduced in [14] and is applied (e.g., [2], [16], [24]) as the measurement of recency for stream data. Based on the experimental studies [7], the exponential decay function has been shown to be effective in blending the recency and text relevancy of objects.

Property: Our scoring method is general and guarantees that the relative ranking of two different objects w.r.t. a query is consistent over time, i.e., if $S_{tsk}(q, o_j, t) > S_{tsk}(q, o_k, t)$, then $\forall \Delta t > 0$ we have $S_{tsk}(q, o_j, t + \Delta t) > S_{tsk}(q, o_k, t + \Delta t)$.

This property indicates that we need not re-rank query results over time. However, the difficulty is that the absolute ranking scores of the objects in top-k results will decrease over time, which may affect the judgment of whether a new object can be one of the results.

In our applications, the typical arrival rate of geo-textual objects (e.g., tweets) is in the scale of millions a day, while new TaSK queries are added at the rate of tens of thousands a day, and we may serve millions of TaSK queries at one time. We thus aim to develop a scalable solution to maintain the up-to-date results for a large number of TaSK queries over a data stream of geo-textual objects. Millions of TaSK queries can easily fit into the available memory of modern servers. Hence, our solution is developed under this setting. In the rare case that the TaSK queries cannot fit into memory, we can employ our proposed solution on multiple servers, each handling a subset of TaSK queries independently.

III. Framework Overview

As mentioned in Section I, an underlying idea of many publish/subscribe systems is to group similar subscription queries such that they can be evaluated simultaneously for a new published object. For example, XFilter [1] employs Non-deterministic Finite Automaton (NFA) to represent multiple XPath subscription queries so that they can be evaluated simultaneously; Shraer et al. [24] propose an approach to grouping text subscription queries such that they can be evaluated simultaneously; In the continuous k nearest neighbor (CkNN) queries [20], each query is represented by a circular influence region with the query location as the center and the distance from the query location to its kth nearest object as the radius, and queries whose influence regions fall in the same spatial region are grouped together so that they can be processed simultaneously.

Motivated by these systems, we also expect to design an approach to grouping TaSK queries such that queries in one group can be evaluated simultaneously, thus reducing the computation of query processing. Moreover, the overhead of updating query groups should be small since TaSK queries arrive and expire as a stream. However, it is challenging to develop such an approach for TaSK queries, each of which has its own location, keywords, and preference parameter (α), and the temporal spatial-keyword score of a current top-k result of a query decreases over time.

To address the challenge, we propose the concept of *Conditional Influence Region* (CIR) to represent each TaSK query, and based on CIRs we develop an efficient mechanism to group queries such that each group can be handled simultaneously. Based on the grouping mechanism, we develop an algorithm for efficiently maintaining the up-to-date top-k results for each TaSK query over geo-textual object stream. Here we index queries instead of geo-textual objects, and run geo-textual objects as queries on that index.

Figure 1 shows our proposed architecture for processing TaSK queries. A user may issue a query or generate a geo-textual object in the system. When our system receives a TaSK query, a complementary index—the *object index* component is used to initialize the top-k result when a new query arrives. The initialization is optional. Next, the query is represented as *CIR* and it is grouped and indexed by the *query index* component, utilizing the component of *cost model for query insertion*. The *query table* maintains the basic information of all queries (including query location, query keywords, query preference parameter α) and their current results with ranking scores. When a new geo-textual object arrives, the query index is utilized to find the queries whose top-k results include the new object, and their top-k results are updated and pushed to the users who issue these queries.

The concept of CIR, the approach to grouping and indexing TaSK queries, the algorithm for processing geo-textual objects on the query index, and the algorithm of query insertion are the techniques we develop in this paper. For object index, we can use any of the existing index structures (e.g., [30]).

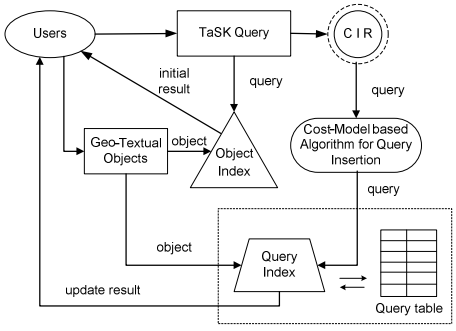


Figure 1. Architecture for Processing TaSK Queries

IV. Representing and Grouping TaSK Queries

We propose a novel approach to representing, grouping, and indexing TaSK queries such that each group of queries can be processed simultaneously for a new geo-textual object. We first propose the concept of Conditional Influence Region (CIR for short) to represent the TaSK query (Section IV-A).

Based on CIRs, we develop an approach to deriving a filtering condition of a TaSK query w.r.t. a spatial region, and show how to use the filtering condition to determine whether a new object falling in the region is a result of the query (Section IV-B). Finally we develop an approach to grouping and indexing TaSK queries and generating filtering conditions for each group of queries (Section IV-C).

A. Conditional Influence Region (CIR)

We propose the concept of Conditional Influence Region (CIR) to represent the TaSK query. The idea of CIR is inspired by the influence region technique for processing continuous k nearest neighbor (CkNN) queries [20] (as discussed in Section III). Each CkNN query is represented by a circular influence region with the query location as the center and the distance from the query location to its kth nearest object as the radius. The influence region plays the role of a *filtering condition*: If a spatial object falls in the influence region of a query, the object becomes a result of the query; otherwise, it cannot be a result.

However, we have no way to generate an influence region for a TaSK query. This is because the ranking score of an object for a TaSK query relies not only on the spatial proximity, but also on the text relevance and the time gap between the object creation time and current time. We propose the conditional influence region (CIR) to help process the TaSK query. Given a query q, we define its CIR to be conditional on: (1) The text relevance score tr; (2) The timestamp t; (3) The current k-th result of q ($R_q[k]$). That is, for query q, we generate CIRs with different radii, each denoted by $\mathsf{CI}_q(tr, t)$. Intuitively, given a new geo-textual object o arriving at time t, of which text relevance to q is tr, o becomes a top-k result of q iff o falls in the CIR $\mathsf{CI}_q(tr, t)$. Table I summarizes the notations frequently used in the rest of this paper.

Definition 3: Conditional Influence Region (CIR). Let $\mathsf{CI}_q(tr, t)$ be the CIR of q w.r.t. time t and text relevance score tr, and $\mathsf{r}_q(tr, t)$ be the radius of $\mathsf{CI}_q(tr, t)$. Based on Definition 2 the relationship among t, tr, $R_q[k]$, and $S_p(\mathsf{r}_q(tr, t))$ can be expressed by Equation 5.

$$S_{tsk}(q, R_q[k], t) = \alpha \cdot S_p(\mathsf{r}_q(tr, t)) + (1 - \alpha) \cdot tr \qquad (5)$$

□

Table I. SUMMARY OF NOTATIONS

Notation	Description
$\mathsf{CI}_q(tr, t)$	CIR of q w.r.t. time t and text relevance score tr
$\mathsf{r}_q(tr, t)$	radius of $\mathsf{CI}_q(tr, t)$
$S_p(\mathsf{r}_q(tr, t))$	the spatial proximity score of $\mathsf{r}_q(tr, t)$
R_q	a list of top-k result of q sorted by S_{tsk}
$R_q[k]$	the k-th result of query q
$R_q[k].t_c$	the creation time of $R_q[k]$
$dist_{min}(q, c)$	the minimum distance between q and cell c
$\mathsf{minD}(q, c)$	spatial proximity score of $dist_{min}(q, c)$
$\mathsf{minT}(q, c)$	minimum conditional text score w.r.t. q and c

From Equation 5, we can see that $\mathsf{r}_q(tr, t)$ increases as the time t goes by, and a larger value of tr results in a larger $\mathsf{r}_q(tr, t)$. Example 2 demonstrates the CIRs given q.

Example 2: Let tr_0 and tr_1 be the text relevance of geo-textual object o_0 and o_1 to query q, respectively, and $tr_0 < tr_1$. Let t_0 and t_1 be two timestamps and $t_0 < t_1$. Figure 2 illustrates the CIRs of query q under different timestamps and different values of text relevance.

We observe that radius becomes larger as time passes and a larger value of tr corresponds to a larger radius. Object o_0 falls outside both $\mathsf{CI}_q(tr_0, t_0)$ and $\mathsf{CI}_q(tr_0, t_1)$. Hence, o_0 is not a result of q at both time t_0 and time t_1. Object o_1 falls outside $\mathsf{CI}_q(tr_1, t_0)$ but inside $\mathsf{CI}_q(tr_1, t_1)$. Therefore, o_1 is a result of q at time t_1 but o_1 is not a result of q at time t_0. □

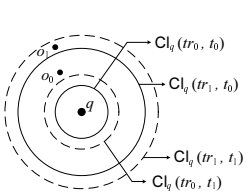
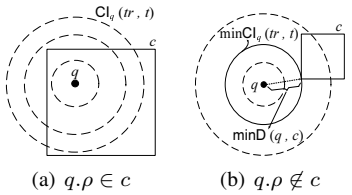


Figure 2. CIRs of q



(a) $q.\rho \in c$ (b) $q.\rho \notin c$

Figure 3. CIRs given q and c

B. Storing a TaSK Query with Filtering Condition

We first propose the method of deriving a *filtering condition* of a TaSK query w.r.t. objects falling in a spatial cell based on the concept of CIR, and then we develop an approach to associating a TaSK query and its filtering conditions with a spatial index. These techniques lay the foundation of our approach to grouping queries and generating a filtering condition of a group of queries (to be presented in Section V).

1) Deriving filtering conditions of a TaSK query: According to Definition 3, the radii of CIRs of a query q depend on: (1) The temporal spatial-keyword score between q and its kth result (in R_q); (2) The text relevance between q and a new geo-textual object o; (3) The arrival time of o. This renders it inapplicable to generate a CIR for q and then use the region as filtering condition to decide whether a new geo-textual object is a result of q (in the similar way as the influence region is used for the CkNN query [20]).

To this end, we propose a novel way of generating CIRs and deriving their corresponding filtering conditions for a TaSK query with respect to a spatial cell. Our idea comprises three steps, which are detailed as follows.

Step 1: Given a spatial cell c and a TaSK query q, we generate the minimum circle, denoting $\mathsf{minCI_q}$, whose radius $dist_{min}(q, c)$ is the minimum distance between q and c. Let $\mathsf{minD}(q, c)$ represent the spatial proximity score of $dist_{min}(q, c)$. Formally, we have

$$\mathsf{minD}(q, c) = S_p(dist_{min}(q, c)). \qquad (6)$$

Note that when q is located in c, the radius denoted by $\mathsf{minD}(q, c)$ is 0; Figure 3(a) illustrates the case when q is located within c and Figure 3(b) illustrates the case when q is not located within c.

Step 2: Based on the circle generated in Step 1, we generate a CIR. Here we know the spatial proximity score of radius ($\mathsf{minD}(q, c)$), the current time t_{cur}, and the current kth result object of q ($R_q[k]$), and we want to compute the current corresponding text relevance score, which is called *minimum conditional text score*.

Definition 4: Minimum Conditional Text Score. Given query q, cell c, the kth result object $R_q[k]$, and $\mathsf{minD}(q, c)$, based on Equation 5, the corresponding text relevance score at current time (t_{cur}) is computed by

$$\mathsf{minT}(q, c) = \frac{S_{tsk}(q, R_q[k], t_{cur})}{1-\alpha} - \frac{\alpha \cdot \mathsf{minD}(q, c)}{1-\alpha}, \quad (7)$$

□

Step 3: This step utilizes the *minimum conditional text score* to check whether a new geo-textual object o is a result for query q if o falls in the spatial area of c. With the scores computed in Definition 4, we have the following lemma to determine whether o is a result.

Lemma 1: Let o be a new geo-textual object located in the spatial cell c. We have: (1) when $TRel(q.\psi, o.\psi) \leq \mathsf{minT}(q, c)$, o is not a result of q; and (2) when $TRel(q.\psi, o.\psi) > \mathsf{minT}(q, c)$, o may be a result of q.

Proof: If $TRel(q.\psi, o.\psi) \leq \mathsf{minT}(q, c)$, then we have $S_{tsk}(q, R_q[k], t_{cur}) \geq (1-\alpha)\, TRel(q.\psi, o.\psi) + \alpha \cdot \mathsf{minD}(q, c)$. Since $\forall o.\rho \in c$, $S_p(dist(q.\rho, o.\rho)) \leq \mathsf{minD}(q, c)$, we have: $S_{tsk}(q, R_q[k], t_{cur}) \geq (1-\alpha)\, TRel(q.\psi, o.\psi) + \alpha \cdot S_p(dist(q.\rho, o.\rho))$. Thus o cannot be a result of q. □

Lemma 1 offers a filtering condition of query q for objects falling in cell c. Specifically, for a new geo-textual object o falling in cell c, we can apply Lemma 1 to check whether each of the queries stored in c will let o become a result.

2) Associating a TaSK *query with a spatial index:* In last subsection, we present an approach to deriving a filtering condition of a TaSK query w.r.t. objects falling in a cell based on CIRs, and the filtering condition enables us to check whether an object falling in the cell is a result of the query. However, a new geo-textual object may fall in any cell in the spatial area. Hence, to utilize Lemma 1 to handle all incoming geo-textual objects, we need to find a set of spatial cells that can cover the whole spatial area, and for each query we maintain its corresponding minimum conditional text score in each cell. Also, we need a spatial indexing structure to organize the cells, queries, and corresponding conditional text scores.

We choose the Quad-tree for the purpose. The reason is that Quad-tree is more suitable for update-intensive applications [11] compared with R-tree based indices, which will incur additional cost for maintaining the MBRs when new queries arrive.

For each query q, we select a set of non-overlapping cells from different levels of the Quad-tree that together cover the whole spatial area; q is associated with each of the selected cells, and for each selected cell c, we generate corresponding $\mathsf{minT}(q, c)$. Figure 4 exemplifies the set of Quad-tree cells (crossed by "X") associated with q_0. The problem of selecting a set of cells in the Quad-tree for a query will be discussed in Section VI.

3) Storing a TaSK *query and its minimum conditional text score:* The next problem is that, given a query q and one of its selected cells c, how to store the minimum conditional text score ($\mathsf{minT}(q, c)$).

Our basic idea is to separately store each component of $\mathsf{minT}(q, c)$ according to: (1) Whether it is time-dependent; (2) Whether it is cell-dependent.

According to Equation 7, $\mathsf{minT}(q, c)$ is computed by the following two parts: (1) $\frac{1}{1-\alpha} S_{tsk}(q, R_q[k], t_{cur}) = \frac{1}{1-\alpha} S_{sk}(q, R_q[k]) \cdot D^{-(t_{cur} - R_q[k].t_c)}$; and (2) $\frac{\alpha}{1-\alpha}\mathsf{minD}(q, c)$.

We note that part (1) is time-dependent but cell-independent, while part (2) is time-independent but cell-dependent. In the rest of the paper, we use $\widetilde{S}(q, R_q[k])$ and $\widetilde{minD}(q, c)$ to denote $\frac{1}{1-\alpha} S_{sk}(q, R_q[k])$ and $\frac{\alpha}{1-\alpha}\mathsf{minD}(q, c)$ respectively. Since part (1) is time-dependent, we separately store $\widetilde{S}(q, R_q[k])$ and $R_q[k].t_c$. Then given t_{cur}, part (1) can be computed through $\widetilde{S}(q, R_q[k])$ and $R_q[k].t_c$. We also maintain two types of query tables: (1) Local query table, which is separately maintained by each Quad-tree cell c, is used for storing the cell-dependent component – $\widetilde{minD}(q, c)$; (2) Global query table, which is used for storing the cell-independent components – $\widetilde{S}(q, R_q[k])$ and $R_q[k].t_c$. Note that the global query table also stores $q.\alpha$, $q.\rho$, and the current results for each q.

Example 3: Consider three TaSK queries, q_0, q_1, and q_2, and a Quad-tree cell c, which is colored as grey in Figure 5, where $q_0.\rho \in c$, $q_1.\rho \notin c$, and $q_2.\rho \notin c$. The information for the three queries, the global query table, and the local query table for cell c are exemplified in Figure 5. □

C. Grouping and indexing TaSK Queries

Based on the techniques in the last two subsections, we propose a new mechanism to group and index TaSK queries in each cell, as well as generate a filtering condition for a group such that the queries in a group can be processed simultaneously.

Our high-level structure in each cell is an inverted file [19]. With the inverted file, for a geo-textual object we only need to consider the queries containing at least one keyword in the object since the object that does not contain any query keyword will not be a result of the query. Nevertheless, it is still inefficient of checking on each posting (query) in the postings lists [19] of each word contained in the object.

To enable group filtering in a postings list for processing queries in a group simultaneously, we propose a technique to group the postings in each postings list into blocks, each of which contains a specified number of postings, and propose an approach to estimating the lower bound of minT for a block. The bound can be used to generate the filtering condition of a block w.r.t. each new object. However, it is non-trivial to derive such a tight bound for a block. The challenge is that the value of minT is determined by each TaSK query, and different queries may have different location, keyword, preference parameter (α), and the top-k result set. To address the challenge, we propose an efficient method for estimating lower bounds of minT, and we develop the *spatial-aware block structure* to optimize the estimation of the lower bounds.

1) Estimating lower bounds of minT*:* Recall that $\mathsf{minT}(q, c)$ is computed by two parts, one is time-dependent and the other is time independent. So we are unable to acquire the exact current minimum $\mathsf{minT}(q, c)$ in b_i unless we compute

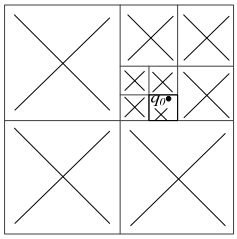


Figure 4. Quad-tree Cells associated with q_0

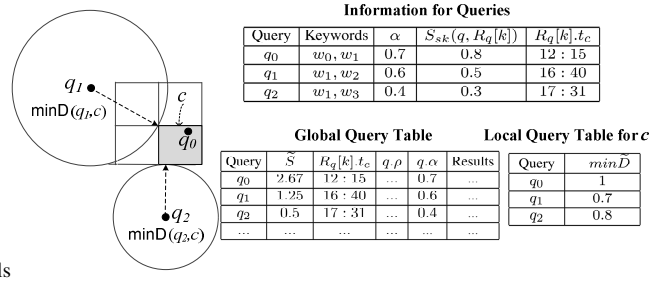


Figure 5. Combining TaSK queries with the Quad-tree

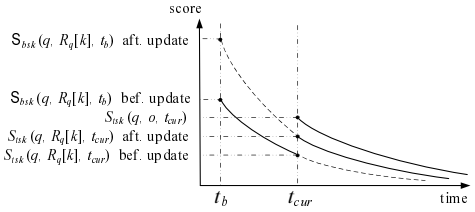


Figure 6. Temporal Effect of the TaSK Query

$\mathsf{minT}(q,c)$ for each $q \in b_i$ at the current time, which is what we want to avoid. Consequently, given a block b_i, we compute the lower bound of $\mathsf{minT}(q,c)$ for all $q \in b_i$ by computing the minimum value of $\frac{1}{1-\alpha}S_{tsk}(q, R_q[k], t_{cur})$ and the maximum value of $\frac{\alpha}{1-\alpha}\mathsf{minD}(q,c)$ respectively for all $q \in b_i$, and combining them based on Equation 7.

However, the challenge here is that different queries in b_i may have different values of $R_q[k].t_c$, which is called the the *base time* of q, and thus we still have to retrieve $R_q[k].t_c$ for all $q \in b_i$ for computing the minimum value of $\frac{1}{1-\alpha}S_{tsk}(q, R_q[k], t_{cur})$ in b_i. To address the problem, our idea is to unify different values of base time of all queries in a block to be a uniform base time, which is denoted by t_b. In particular, for each query q we compute the equivalent spatial-keyword score of $S_{sk}(q, R_q[k])$ at the base time t_b, rather than at the creation time of $R_q[k]$ ($R_q[k].t_c$). We next introduce how to compute equivalent spatial-keyword score.

Definition 5: Equivalent Spatial-Keyword Score. Let q be a TaSK query, the equivalent spatial-keyword score between q and $R_q[k]$ at time t_b is computed by:

$$\mathsf{S}_{bsk}(q, R_q[k], t_b) = S_{sk}(q, R_q[k]) \cdot D^{R_q[k].t_c - t_b} \quad (8)$$

□

Example 4: Figure 6 illustrates the variation of $S_{tsk}(q, R_q[k], t_{cur})$ w.r.t. the current time. t_{cur}^- denotes the timestamp just before the current time and t_{cur}^+ represents the timestamp just after the current time. t_b is the unified base time. $\mathsf{S}_{bsk}(q, R_q[k], t_b)$ is the equivalent spatial-keyword score for q at t_b. When a new object o arrives at t_{cur}, we use $\mathsf{S}_{bsk}(q, R_q[k], t_b)$ to compute $S_{tsk}(q, R_q[k], t_{cur})$ as follows according to Definition 5:

$$S_{tsk}(q, R_q[k], t_{cur}) = \mathsf{S}_{bsk}(q, R_q[k], t_b) \cdot D^{-(t_{cur} - t_b)}$$

Then we find that $S_{tsk}(q, o, t_{cur}) > S_{tsk}(q, R_q[k], t_{cur})$, and thus o becomes a result of q and we update $\mathsf{S}_{bsk}(q, R_q[k], t_b)$. □

Based on Equations 7 and 8, we compute $\mathsf{minT}(q,c)$ as follows:

$$\mathsf{minT}(q,c) = \frac{\mathsf{S}_{bsk}(q, R_q[k], t_b) \cdot D^{-(t_{cur} - t_b)}}{1-\alpha} - \frac{\alpha \cdot \mathsf{minD}(q,c)}{1-\alpha},$$

Hence, we have the following lemma for estimating the lower bound of $\mathsf{minT}(q,c)$.

Lemma 2: Given a block b_i, let $b_i.minT$ be the minimum value of $\mathsf{minT}(q,c)$ for all $q \in b_i$, $b_i.minS$ be the minimum value of $\frac{1}{1-\alpha}\mathsf{S}_{bsk}(q, R_q[k], t_b)$ for all $q \in b_i$, and $b_i.minD$ be the maximum value of $min\widetilde{D}(q,c)$ for all $q \in b_i$, then $b_i.minS \cdot D^{-(t_{cur} - t_b)} - b_i.minD$ can be the lower bound of $b_i.minT$. Specifically, we have:

$$b_i.minT \geq b_i.minS \cdot D^{-(t_{cur} - t_b)} - b_i.minD \quad (9)$$

Proof: Given a block b_i, assume that $q_m \in b_i$ s.t. $\forall q \in b_i$, $\mathsf{minT}(q_m, c) \leq \mathsf{minT}(q, c)$. We have

$$b_i.minT = \frac{1}{1-\alpha}\mathsf{S}_{bsk}(q_m, R_{q_m}[k], t_b) \cdot D^{-(t_{cur} - t_b)} - min\widetilde{D}(q_m, c).$$

Since $\frac{1}{1-\alpha}\mathsf{S}_{bsk}(q_m, R_{q_m}[k], t_b) \geq b_i.minS$ and $min\widetilde{D}(q_m, c) \leq b_i.minD$, we have Equation 9. □

2) Spatial-aware block structure: We proceed to present our proposed spatial-aware block structure for organizing postings lists. To filter as many blocks as possible in each postings list when a new geo-textual object arrives, we want to acquire a relatively tight lower bound for each block. If we can organize queries such that queries in a block have similar values of minT, it is more likely that we can estimate a tighter lower bound for minT. To achieve this, we partition the queries associated with cell c based on their corresponding $min\widetilde{D}(q,c)$, which is invariable w.r.t. both the time and the kth result $R_q[k]$. Specifically, in each cell c queries are first partitioned by the $min\widetilde{D}$-buckets (Definition 6), then queries in each bucket are indexed by a block based inverted file.

Definition 6: $min\widetilde{D}$-Bucket. We use $min\widetilde{D}$-buckets to organize blocks in a cell. Each $min\widetilde{D}$-bucket ("bucket" for short) corresponds to a score interval of $min\widetilde{D}$, and is associated with an order number r. Namely, the rth bucket B_r corresponds to the following interval: $[r \cdot \Delta D, (r+1) \cdot \Delta D)$, where ΔD is a specified parameter indicating the range of $min\widetilde{D}$ in a bucket. A query q is in bucket B_r if: $r \cdot \Delta D \leq min\widetilde{D}(q,c) < (r+1) \cdot \Delta D$. □

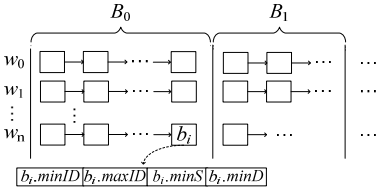


Figure 7. Structure of Spatial-Aware Block Based Inverted File

Figure 7 illustrates the structure of the spatial-aware block based inverted file. The TaSK queries are firstly partitioned into a pre-specified number of buckets based on their $min\widetilde{D}(q,c)$. For each bucket B_i, a separate block based inverted file is maintained where the queries in each list are sorted according to their query ids in ascending order. Each block b_i contains at most p_{max} queries, where p_{max} is a specified parameter. Each posting of a query just stores its

query id. To compute the lower bound based on Lemma 2, we augment each block b_i with the following values: (1) $b_i.minID$ and $b_i.maxID$, which respectively indicate the minimum and maximum ids of queries in b_i; (2) $b_i.minS$, which is the minimum value of $\frac{1}{1-\alpha}\mathsf{S}_{bsk}(q, R_q[k], t_b)$ for all $q \in b_i$; (3) $b_i.minD$, which is the maximum value of $min\widetilde{D}(q, c)$ for all $q \in b_i$ (the maximum value of $min\widetilde{D}(q, c)$ corresponds to the minimum distance between q and c).

V. Algorithm for Processing TaSK Queries

We proceed to present the algorithm for processing and updating indexed TaSK queries. Recall that each query q is associated with a set of non-overlapping cells that cover the whole space. Hence given a query q, for any location ρ in the space there must exist a cell c s.t. $\rho \in c$ and c contains q's posting. Consequently, when a new geo-textual object o arrives, we only traverse the inverted file in the cells whose areas cover $o.\rho$.

Based on the filtering condition of a block w.r.t. each new object, we propose an algorithm for traversing the inverted file with forward skipping. Our high-level idea is as follows. For a cell containing object o we traverse its postings lists of all the words contained in o simultaneously based on the Document-at-a-Time (DAAT) technique [19]; for each postings list we maintain a cursor that specifies the query id we currently visit. Since the query ids are sorted in ascending order within each $min\widetilde{D}$-bucket, the queries whose ids are smaller than the id at the cursor position of the postings list of keyword w may belong to two cases: (1) they have been evaluated; or (2) they do not contain w. For each block b_i from the postings lists of words in object o, we generate a filtering condition (Lemma 3) to check whether b_i can be safely filtered without evaluating each individual query in b_i.

Lemma 3: Let o be a geo-textual object, b_i be a block in bucket B in the postings list of keyword w' ($w' \in o.\psi$), and $S^o_{b_i}$ be the subset of $o.\psi$ such that $S^o_{b_i} = \{w | w \in o.\psi \wedge p^w_{c,B} \leq b_i.maxID\}$ where $p^w_{c,B}$ denotes the current position of the cursor in the postings list of w in Bucket B under cell c. We use $TRel_{max}(o.\psi, S^o_{b_i})$ to denote the maximum possible text relevance between $o.\psi$ and the queries containing keywords in $S^o_{b_i}$. Then we have the following proposition: If $b_i.minT \geq TRel_{max}(o.\psi, S^o_{b_i})$, o will not be a result of any query in b_i.

Proof: Since the query ids in B are sorted in ascending order, if $p^{w'}_{c,B} > b_i.maxID$, then based on the DAAT scheme it suggests that: (1) o cannot be a result of any query in b_i; or (2) queries in b_i that can match o have already been found. Hence, unevaluated queries in b_i cannot contain any keyword that does not belong to $S^o_{b_i}$. So according to Equation 3 the text relevance between o and any query in b_i cannot exceed $TRel_{max}(o.\psi, S^o_{b_i})$. Then based on Lemmas 1 and 2, if $b_i.minT \geq TRel_{max}(o.\psi, S^o_{b_i})$, o cannot be a result of any query in b_i. □

Lemma 3 offers a filtering condition for efficiently determining whether object o is not a result of any query in a block. If it can be determined, we move the cursor to the first query in the next block; otherwise the cursor is forwarded to the next query, and we need to check each individual query in the block to determine whether o is a result.

Algorithm 1: ObjectProcess(**Object** o)

```
1  Result ← ∅;
2  c ← CIQ.root;
3  while cell c is not empty do
4      for each bucket B_i in c do
5          S_w ← o.ψ;
6          ∀w ∈ S_w, p^w_{c,B_i} ← id of the first query in
           I(w, c, B_i);
7          while S_w ≠ ∅ do
8              w_m ← term with the minimum p^w_{c,B_i} for all
               w ∈ S_w;
9              p^{w_m}_{c,B_i} ← FindNext(w_m, {p^w_{c,B_i}}_{w∈S_w}, Result);
10             if p^{w_m}_{c,B_i} reaches the end of the current block b_c
               then
11                 update b_c.minS;
12             if p^{w_m}_{c,B_i} reaches the end of I(w_m, c, B_i) then
13                 S_w ← S_w \ w_m;
14     c ← c's child node that contains o.ρ;
```

We are now ready to present our algorithm for processing an incoming object o to maintain the top-k results of individual queries. The algorithm starts from the root cell of the Quad-tree and iteratively visits the child cells that cover the object o until we reach the leaf cell; At each visited cell, queries in each $min\widetilde{D}$-bucket B_i are separately evaluated. For evaluating the queries in each bucket, it traverses the postings lists for each word in o concurrently, and applies Lemma 3 to prune the search space.

Algorithm 1 shows the pseudo code. For each bucket B_i in a visited cell, we first initialize the cursor of each postings list to be the first element in the list (lines 5–6). Here $I(w, c, B_i)$ represents the postings list for keyword w in bucket B_i under cell c (line 6). Next, we choose the postings list of w_{min} where its cursor is located at the posting with the minimum query id among all the postings lists of the words of o (line 8). Then we invoke function FindNext to evaluate the blocks/queries sequentially until w_{min} is not the keyword with the minimum p^w_{c,B_i} for all $w \in S_w$, where S_w represents the keyword set s.t. $(\forall w_j \in S_w)\ p^{w_j}_{c,B_i} < I(w_j, c, B_i).maxID$ (line 9). If the cursor reaches the end of the current block b_c, we update $b_c.minS$ (line 10). We will remove w_{min} if $p^{w_{min}}_{c,B_i}$ reaches the end of $I(w_{min}, c, B_i)$ (lines 12–13). When S_w is empty, we move to the next bucket.

Function FindNext($w, \{p^w_{c,B_i}\}_{w\in S_w}$, *Result*)

```
1  p_c ← p^w_{c,B};
2  b_c ← the block containing p_c;
3  if p_c = b_c.first then
4      if b_c.minT ≥ TRel_max(o.ψ, S^o_{b_c}) then
5          b_c ← b_c.next; p_c ← b_c.first;          // Lemma 3
6  else
7      q ← the query indicated by p_c;
8      if TRel(o.ψ, q.ψ) > minT(q, c) then
9          Compute S_tsk(q, o, t_cur);
10         if o is the result of q then
11             Return q as the query that matches o;
12             Update the the result of q in the global query list;
13     p_c ← p_c.next;
14 return p_c;
```

Function FindNext is used for checking whether object o is a top-k result of the current block/query. First, p_c and b_c are respectively initialized as the current posting and block (lines 1–2). If p_c is the first posting in b_c, i.e., none of the queries in b_c have been evaluated, then we compare $b_c.minT$ with $TRel_{max}(o.\psi, S^o_{b_c})$. If $b_c.minT \geq TRel_{max}(o.\psi, S^o_{b_c})$, none of the queries in b_c can have o as a result, and thus we skip b_c and forward the cursor to the next block (lines 4–6). However, if p_c is not the first posting in b_c, which indicates that b_c cannot be skipped as a whole, then we need to consider the following two cases: (1) If the text relevance between o and q ($TRel(o.\psi, q.\psi)$) is larger than $\mathsf{minT}(q, c)$, we compute $S_{tsk}(q, o, t_{cur})$ and check whether o is a result of q. If so, q is returned and we update the result of q in the global query list (lines 9–13). (2) If $TRel(o.\psi, q.\psi)$ is smaller than $\mathsf{minT}(q, c)$, then o cannot be a result of q, and we just skip q.

VI. Query Insertion

Recall that each TaSK query needs to be associated with a set of non-overlapping spatial cells that can cover the whole spatial area. To choose such a set of cells for a better performance, we propose a cost model based approach.

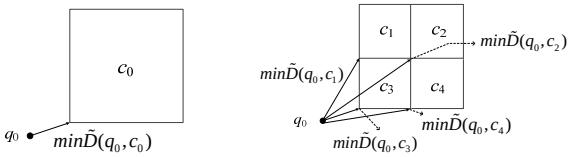
Heuristic Method for Query Insertion: Before introducing our proposed method, we first give a heuristic method for associating a new query onto the Quad-tree cells:

Step 1: Starting from the root cell c_{root} of the Quad-tree, we check whether each of c_{root}'s children c_i covers $q.\rho$. If not, we associate q with c_i. If so, we invoke *Step* 2 with c_i as the input.

Step 2: Given an input cell c, we check whether each of c's children c_j covers $q.\rho$. If not, we associate q with c_j. If so, we recursively invoke *Step* 2 with c_j as the input.

Such recursive procedure terminates when the input cell c in *Step* 2 reaches the m-th layer of the Quad-tree, where m is a tunable parameter. An example association by this heuristic method is given in Figure 4, and this method is denoted by CIQ-H (Heuristic CIR based Quad-tree).

Next we present our cost-model based method denoted by CIQ (CIR based Quad-tree). The objective for selecting a set of cells to associate a new query q is twofold: (1) Minimize the number of evaluations of q, which occurs when q cannot be skipped according to Lemma 1 while processing a geo-textual object; (2) Decrease the number of cells used for associating q, which incurs cost. However, we need to strike a balance between the two objectives that have a trade-off.



(a) $minD$ on c_0 (b) $minD$s on c_0's children

Figure 8. $minD$s on cells from different layers

A. Cost Estimation

Because we do not know the text and location information of future geo-textual objects, it is impossible to compute the exact number of evaluations for query q, So we estimate the expected number of evaluations of q by Definition 7.

Definition 7: Expected Number of Evaluations: Given query q and cell c, F_o denotes the arrival frequency of geo-textual objects, $P(q.\psi)$ denotes the probability of an object containing any of the keywords in $q.\psi$, $P(c)$ is the probability of $o.\rho \in c$, $E(\Delta t)$ indicates the expected duration of q (from the creation time to the deletion time), and $P(TRel > \mathsf{minT}(q, c))$ denotes the probability of $TRel(q, o) > \mathsf{minT}(q, c)$. If q is associated with cell c, the expected number of evaluations for q is computed by

$$E\#(q, c) = F_o \cdot P(q.\psi) \cdot P(c) \cdot P(TRel > \mathsf{minT}(q, c)) \cdot E(\Delta t).$$

□

We estimate F_o, $P(c)$, $P(q.\psi)$, and $E(\Delta t)$ based on historical data, which is straightforward. We estimate $P(TRel > \mathsf{minT}(q, c))$ through the probability density function for the text relevance between an arbitrary query and an arbitrary object that contains at least one query keyword.

Next, we illustrate the comparison between associating q with c_0 and associating q with c_0's children (c_1, c_2, c_3, and c_4 in Figure 8). If we associate q with c_0, then $\mathsf{minT}(q, c_0) = \frac{1}{1-\alpha}\mathsf{S}_{bsk}(q, R_q[k], t_b) - \widetilde{minD}(q, c_0)$ and it will be used for checking new objects falling in c_0. If we associate q with c_0's children, for each $c_i \in c_0.children$ we have $\mathsf{minT}(q, c_i) = \frac{1}{1-\alpha}\mathsf{S}_{bsk}(q, R_q[k], t_b) - \widetilde{minD}(q, c_i)$, which will be used for checking incoming objects falling in c_i.

Lemma 4: Given a quad-tree cell c and a query q, we have

$$E_\#(q, c) \geq \sum_{c_i \in c.children} E_\#(q, c_i), \tag{10}$$

Proof: Since $(\forall c_i \in c.children)\ \mathsf{minD}(q, c) \geq \mathsf{minD}(q, c_i)$, $(\forall c_i \in c.children)\ \mathsf{minT}(q, c) \leq \mathsf{minT}(q, c_i)$. We know that $P(c) = \sum_{c_i \in c.children} P(c_i)$. Hence, we have $E_\#(q, c) \geq \sum_{c_i \in c.children} E_\#(q, c_i)$ based on Definition 7. □

According to Lemma 4, associating query q with the child cells of c will lead to smaller value of $E_\#$ for the incoming objects falling in the area of c. However, both the space and the time cost for storing q will increase as the number of cells used for associating q mounts up. Hence, we need to balance the trade-off between $E_\#$ and the cost for storing q.

To determine whether q is to be associated with cell c or its child cells, we compute the difference of their expected number of evaluations based on Lemma 4 as follows:

$$\Delta E_\#(q, c) = E_\#(q, c) - \sum_{c_i \in c.children} E_\#(q, c_i). \tag{11}$$

Then we compute the difference of the number of insert operations between the two options as follows:

$$\Delta U_\#(q, c) = 3 \times |q.\psi|. \tag{12}$$

To make the two types of cost comparable, we introduce a cost normalization factor δ that specifies the weight between the two types of cost. If $\Delta E_\#(q, c) > \Delta U_\#(q, c) \times \delta$, i.e., the benefit from associating the query with the child cells outperforms the overhead, then we associate q with the child cells of c; otherwise we associate q with c. The factor δ is empirically set at 3.5 in our experiments.

B. Inserting a TaSK Query

The method for query insertion is summarized as follows. It starts from the root cell of the Quad-tree and recursively checks whether we associate the query with the current cell or its child cells. Algorithm 2 shows the pseudo code for query insertion. Specifically, if $\Delta E_{\#}(q,c) \leq \Delta U_{\#}(q,c) \times \delta$, the query will be associated with the current cell c; Otherwise we need to further check whether q will be associated with c's child nodes or lower level descendant nodes by invoking the comparison again for each child of c.

Algorithm 2: QueryInsertion(**Query** q**, Cell** c**, Factor** δ)

```
1 if ΔE_#(q,c) ≤ ΔU_#(q,c) × δ then
2 |   Store the postings of q into c;
3 else
4 |   for each c_i ∈ c.children do
5 |   |   QueryInsertion(q, c_i, δ);
```

After we select a set of cells for associating query q, for each selected cell c, we map q to a bucket based on $\mathsf{minD}(q,c)$. Then for each keyword in q, we insert q into the postings list of the keyword in the bucket. Specifically, we maintain a temporary block b_u for each postings list under each bucket, which serves as a "buffer" to receive the newly inserted queries. If b_u becomes full, we compute $b_u.minID$, $b_u.maxID$, $b_u.minS$, and $b_u.minD$. Subsequently, we insert b_u into the corresponding postings list and then create a new temporary block to replace b_u.

Query Deletion: The operation for query deletion is conducted during query processing. When a deletion request for query q is received, we mark q as expired in the global query list. If we find that all the queries in a block b_i are expired during the process of evaluating queries in b_i, then we delete b_i.

VII. Experimental Study

A. Baselines

We discuss how to exploit existing techniques for processing TaSK queries. No algorithm exist for solving the problem. We develop two baselines by utilizing existing index structures.

1) Inverted File plus Query List (IFL): We use the inverted file to index TaSK queries. We also maintain a global query table. For each query q, the table stores the query id, $S_{bsk}(q, R_q[k], t_b)$, $q.\rho$, $q.\alpha$, and its current query results. When a geo-textual object o arrives, we traverse the postings lists associated with the words in the object in the Document-at-a-Time (DAAT) [19] manner. For each posting, we get the query information from the query table for computing the temporal spatial-keyword score between o and the query ($S_{tsk}(q,o,t_{cur})$). Then we compare $S_{tsk}(q,o,t_{cur})$ with $S_{tsk}(q,R_q[k],t_{cur})$, which is computed from $\mathsf{S}_{bsk}(q,R_q[k],t_b)$, to determine whether o is a result of q. If so, we update the result set of q with o in the query table.

2) Block based Inverted File (BIF): This baseline uses a block based inverted file (BIF) to index the TaSK queries. Similar to the CIR based Quad-tree (CIQ), BIF partitions each postings list into blocks, each of which contains a pre-specified number of postings. For each block b_i, we maintain $b_i.minID$, $b_i.maxID$, and $b_i.minS$. Similar to Algorithm 1, while processing each new object o we traverse the corresponding postings lists with forward block skipping technique. Since BIF does not utilize any spatial partitioning scheme, we cannot derive $\mathsf{minD}(q,c)$ for each query. Hence, while computing the minimum value of $\mathsf{minT}(q,c)$ for each block b_i, we use "1", which is the maximum possible value of the distance score, to replace $\mathsf{minD}(q,c)$ in Equation 9.

Table II. Default Values for each Parameter

Parameter	Default Setting
number of query keywords	TWE: random from 1 to 5
preference parameter α	random from 0 to 1
number of maintained query results	random from 10 to 30
number of postings in each block	FSQ:128 TWE:1024
number of $minD$-buckets for each cell	16
decaying scale	0.5
cost normalization factor δ	3.5
number of indexed queries	FSQ:1M TWE: 10M

B. Experimental Settings

Our experiments are conducted on two real datasets: FSQ and TWE[1]. FSQ is a real-life dataset collected from Foursquare, which contains 1.1 million worldwide POIs with both location and text. The dataset TWE is a larger real-life dataset that comprises 40 million tweets with geo-locations.

The TaSK queries are generated as follows. For FSQ, each POI is mapped to a TaSK query, in which the text description of the POI becomes the query keywords, and the location of the POI becomes the query location. For TWE, we randomly select a number of keywords from the tweet keywords, and the query location is the same as the tweet location. In addition, for experiments on FSQ, we regard each tweet with geo-location from TWE as a geo-textual object, and we use those tweets to annotate the POIs from FSQ. For experiments on TWE, each tweet with geo-location from TWE is considered to be a geo-textual object on TWE. Default value for parameters is presented in Table II.

We implemented all algorithms in Java on a PC with Intel(R) Core(TM) i7-4770k @3.50GHz and 16GB RAM.

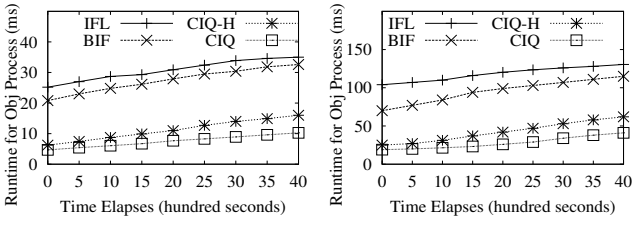
C. Experimental Result

1) the Time Effect: In this set of experiments, each method runs for 4,000 seconds (which is simulation duration, denoted by Δt_{sim}) on both FSQ and TWE. We set the decaying scale $D^{-\Delta t_{sim}}$ at 0.5. Our proposed method is denoted by CIQ, and the variation of CIQ (without the cost-model based method for query insertion) is denoted by CIQ-H as presented in Section VI. The arrival rate of geo-tagged tweets in Twitter is around 100 (4,600 tweets/second [18] and 2.17% of the tweets are geo-tagged[2]). To make sure that all the methods can handle, we use the following setting: during each second 5 geo-textual objects are issued, 5 new queries are issued, and 5 queries become expired. At the beginning, each method is initialized with 1M and 10M TaSK queries, respectively, for FSQ and TWE.

We report the average runtime for processing an object and the average runtime for inserting a query during each period

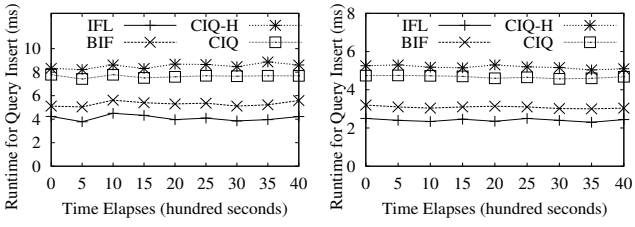
[1]Datasets are available at: http://www.ntu.edu.sg/home/gaocong/datacode.htm

[2]http://journalistsresource.org/studies/society/social-media/mapping-global-twitter-heartbeat-geography#



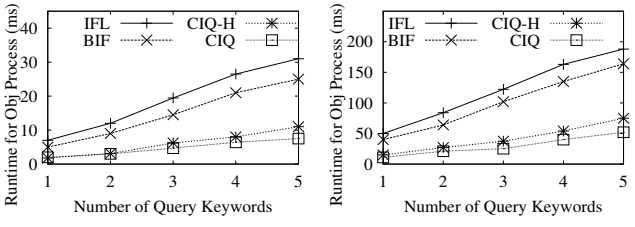
(a) FSQ (b) TWE

Figure 9. Effect of Time for Object Processing



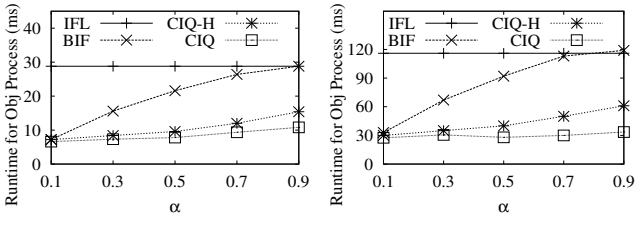
(a) FSQ (b) TWE

Figure 10. Effect of Time for Query Insertion



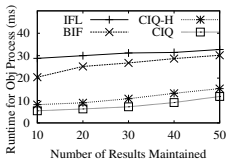
(a) FSQ (b) TWE

Figure 11. Effect of the Number of Query Keywords



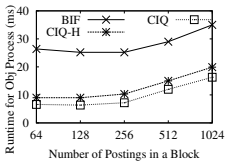
(a) FSQ (b) TWE

Figure 12. Effect of Preference Parameter (α)



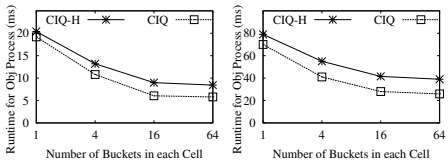
(a) FSQ (b) TWE

Figure 13. Effect of # Query Results



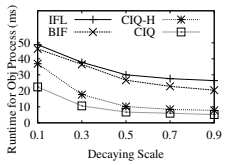
(a) FSQ (b) TWE

Figure 14. Effect of # Postings in a Block



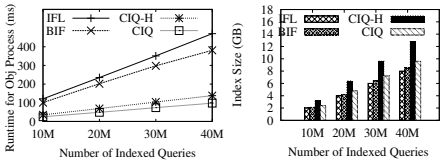
(a) FSQ (b) TWE

Figure 15. Effect of # Buckets in each Cell



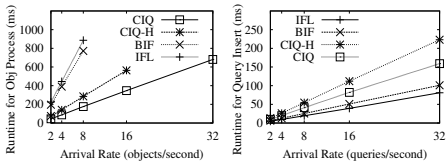
(a) FSQ (b) TWE

Figure 16. Effect of the Decaying Scale



(a) Runtime (b) Index Size

Figure 17. Effect of # Indexed Queries



(a) Object Arrival Rate (b) Query Arrival Rate

Figure 18. Effect of Arrival Rate

of 500 seconds. Figure 9 shows that both CIQ-H and CIQ outperform the two baselines significantly in object processing, and CIQ exhibits the best performance. CIQ is able to improve the runtime performance of the best baseline BIF by 70%-80% on both datasets. The reasons could be explained as follows.

For IFL, we need to check each posting in the postings list of each word of the incoming geo-textual object. While for BIF, postings are indexed by blocks, which may help prune the queries in a block-based manner during the search of postings list. As a result, BIF performs moderately better than IFL. However, compared with CIQ-H and CIQ, BIF does not include the spatial information of each query while building the index, and thus its pruning technique is not able to consider the spatial proximity between the queries and incoming objects. Consequently, CIQ-H and CIQ perform substantially better than BIF. In addition, CIQ improves the runtime performance of CIQ-H by 10% to 30%. Such performance improvement is contributed by the cost-model based method for associating a new TaSK query onto the cells of the Quad-tree.

Figure 10 shows the performance of query insertion for each structure. Since CIQ-H and CIQ require the query to be stored in the postings list under each cell associated with the query, the runtime costs of query insertion for CIQ-H and CIQ are higher than IFL and BIF. However, the time for processing updates is negligible compared with the time for object processing by comparing the runtime in Figure 9. Here the frequency of object processing and the frequency of query insertion are the same. In the publish/subscribe scenario the frequency of query insert is normally much lower than that of object arrivals, and thus the portion of update cost will be even smaller than that shown here. Hence, in the rest of experiments, we only show the object processing cost while ignoring the update cost. Note that the deletion of queries is performed together with object processing as mentioned in Section VI and the time is included in that for object processing.

2) the Number of Query Keywords: We proceed to evaluate the effect of the number of keywords of each TaSK query. Figure 11 shows that all the methods present an increasing trend for the runtime of object processing as we increase the number of query keywords. This is because the number of query keywords is proportional to the number of postings required for indexing the query, which will lead to an increase in the length of each postings list. We also observe that CIQ is able to improve on the runtime of BIF by 60%-70%.

3) Effect of α: In this experiment, we investigate the effect of the query preference parameter α. We observe similar trends on both datasets. For IFL, the performance of object processing is not affected by α. The reason is that IFL needs to retrieve and evaluate all the postings in each postings list of the object keyword while processing an incoming object, irrespective of

the value of α. Hence the value of α will not affect the performance of IFL. However for BIF, the pruning strategy for traversing the postings lists is based on the text relevance. Consequently, more emphasis on the spatial aspect will lower the effectiveness of the text-based pruning strategy. As for CIQ and CIQ-H, the pruning strategy is based on both spatial and text aspects. We observe that as α becomes larger, the disparity between CIQ and BIF becomes larger. At α = 0.9 (distance score has a high weight), CIQ improves over BIF by 70% on FSQ and 75% on TWE. When α = 0.1, the performances of CIQ and BIF are close.

4) the Number of Maintained Query Results: This experiment evaluates the performance w.r.t. parameter "$q.k$". Figure 13 shows that the runtime for object processing slightly increases as we increase the value of $q.k$ for each query. The reason is that the higher value of $q.k$ will induce the lower value of the temporal spatial-keyword score between an incoming object and the kth result maintained for each query on average. Consequently, the average number of queries that have o as a result will increase.

5) the Number of Postings in each Block: This round of experiments is to evaluate the performance of the indices utilizing block structure, including CIQ, CIQ-H, and BIF, when we vary the block size. Figure 14 shows the trend of the object processing cost w.r.t. the block size. On FSQ, all the three indices perform best when each block contains 128 postings, while for TWE the value is changed to 1024. This can be explained as follows. If the block size is too small, then the number of blocks we need to evaluate will increase. On the other hand, if the block size is too large, the possibility for skipping a block will decrease despite the reduction of the number of blocks to be visited while processing an incoming object. Nevertheless, the performance is not significantly affected by the block size for all indices.

6) the Number of Buckets in each Cell: In this experiment, we vary the number of buckets in each cell for CIQ-H and CIQ. Figure 15 shows that both methods exhibit better performance as we increase the number of buckets in a cell. However, when we increase the number of buckets from 16 to 64, the improvement is insignificant on both datasets.

7) the Decaying Scale: Figure 16 shows that the runtime for object processing decreases as we increase the decaying scale. The reason is that a lower value of decaying scale will increase the number of queries that have an incoming object to be their results.

8) the Number of Indexed Queries: We evaluate the effect of the number of indexed queries. The number of queries scales from 10M to 40M. Obviously, increasing the number of indexed queries leads to the increase of postings in each postings list. Hence, more postings will be retrieved and evaluated while processing a new object. Figure 18 shows that both the runtime for object processing and the index size exhibit a linearly increasing trend for all methods as we increase the number of indexed queries. Note that for CIQ the query tables (including both global query tables and local query tables) take 70% to 75% of the total memory cost. We can see that memory cost would not be an issue considering the available memory of PC.

9) Arrival Rate: We vary the arrival rates of both objects and queries. Figure 18(a) presents the total time costs in every 1 second for object processing when we vary the arrival rate of geo-textual objects from 2 to 32 object(s)/second with 10M TaSK queries indexed. We find that CIQ is capable of processing 32 geo-textual objects with 10M indexed queries while the other methods fail to handle.

Figure 18(b) presents the total runtime of query insertion when we vary the arrival rate of TaSK queries. Although the query insertion cost of CIQ is moderately higher than the two baselines, the arrival rate of query is much lower than the arrival rate of object under the publish/subscribe scenario.

VIII. Related work

Continuous kNN Queries. Our problem is related to the problem of continuously monitoring spatial kNN queries over moving objects, which monitors the nearest k objects to a given query point among all the moving objects. Yu et al. [28] and Xiong et al. [27] study the problem of periodically updating results for continuous kNN queries over moving objects. Mouratidis et al. [20] propose a method for evaluating continuous kNN queries based on a grid index and the concept of influence region. These proposals focus on moving objects while our work aims to handle a stream of geo-textual objects. We do not see a way to adapt them for handling TaSK queries.

Top-k Spatial Keyword Search. Top-k kNN Query (TkQ) returns k most spatial-textual relevant objects that are ranked by both spatial proximity and text relevancy. Several geo-textual indices have been proposed to efficiently answer TkQ, such as the IR^2-tree [10], the IR-tree [6], S2I [22], I^3 [30], and IL-Quadtree [29]. Among them, the IR^2-tree [10], the IR-tree [6] and S2I [22] are based on the R-tree, and the others are based on the Quad-tree. There exits no sensible way to adopt these methods to handle TaSK queries.

Reverse kNN Search. The TaSK query can be viewed as for each incoming geo-textual object finding the set of queries that take the geo-textual object as one of their top-k results according to the temporal spatial-textual scores. In this sense, the TaSK query is also related to the Reverse k Nearest Neighbor (RkNN) query, which is to find the set of objects that take a query as one of their kNN based on the spatial distance. The RkNN query has been studied extensively (e.g., [3], [12], [25]). The textual relevance is also considered for the reverse kNN query [17]. However, these techniques cannot be used for answering the TaSK query because they just consider the one-time query over static objects.

Content based Publish/Subscribe. Closest to our problem setting is the existing work on top-k publish/subscribe systems [8], [9], [21], [24] that make published items trigger a subscription only if it ranks among the top-k published items. In the setting of most of these systems [8], [9], [21], the relevance of an item remains constant during a pre-specified time interval, and once its lifetime exceeds the item simply expires. The expired item is then replaced by the most relevant unexpired item. The setting is different from our setting where time is part of the ranking score. The setting of top-k publish/subscribe system [24] is similar to ours, where the published items are tweets and the subscriptions are news. The published items (e.g., tweets) do not have a fixed expiration

time. Instead, time is a part of the relevance score, which decays as time passes. Older items retire from the top-k only when new items that score higher arrive. The inverted files are used as the indexes and the classic information retrieval methods are adapted for the ranking. Our work differs from this study in that both queries and objects in our work are geo-textual. The spatial aspect is part of the ranking score, which renders the solution [24] inapplicable, and also introduces new challenges for top-k publish/subscribe.

Several publish/subscribe systems [4], [13], [26] are developed for geo-textual objects. To index subscription queries, Chen et al. [4] present a hybrid index based on Quad-tree and Inverted files, and Li et al. [13] present a hybrid index based on R-tree and Inverted files. Recently, Wang et al. [26] propose a novel adaptive spatial-textual partition tree that adaptively groups the subscription queries based on keyword and spatial partitions, guided by a cost model. However, their publish/subscribe problem is different from the top-k publish/subscribe problem, and their methods cannot be employed to handle TaSK queries.

News Detection over Tweets Stream. Sankaranarayanan et al. [23] develop a news processing system *TwitterStand* to continuously acquire breaking news from the tweets generated by some selected users ("Seeders") who publish news. Their published tweets are clustered together, and each cluster of tweets is associated with a set of geographical locations by analyzing the tweet content and tweet meta-data. Users can specify the topics and geographical regions of interest, and summaries of clusters w.r.t. the specified topics and regions are displayed on the map. However, the queries in *TwitterStand* are continuous spatial-keyword queries with boolean filtering expressions, which are different from our TaSK queries. Moreover, the work does not consider how to efficiently process a large number of subscription queries.

IX. Conclusion

We consider the problem of maintaining up-to-date results for a large number of TaSK queries that take into account text relevance, spatial proximity, and recency of geo-textual objects. We propose a mechanism to efficiently processing a large number of TaSK queries. In particular, based on the concept of conditional influence region, we develop an approach to grouping and indexing TaSK queries and generating filtering conditions for each group of queries to evaluate them simultaneously. The experimental results on two real-world datasets show that our solution is able to achieve a reduction of the processing time by 70–80% compared with two baselines.

Acknowledgment

This work is supported in part by a grant awarded by a Singapore MOE AcRF Tier 2 Grant (ARC30/12).

References

[1] M. Altinel and M. J. Franklin. Efficient filtering of xml documents for selective dissemination of information. In *VLDB*, pages 53–64, 2000.

[2] G. Amati, G. Amodeo, and C. Gaibisso. Survival analysis for freshness in microblogging search. In *CIKM*, pages 2483–2486. ACM, 2012.

[3] M. A. Cheema, X. Lin, W. Zhang, and Y. Zhang. Influence zone: Efficiently processing reverse k nearest neighbors queries. In *ICDE*, pages 577–588, 2011.

[4] L. Chen, G. Cong, and X. Cao. An efficient query indexing mechanism for filtering geo-textual data. In *SIGMOD*, pages 749–760, 2013.

[5] L. Chen, G. Cong, C. S. Jensen, and D. Wu. Spatial keyword query processing: an experimental evaluation. In *PVLDB*, pages 217–228, 2013.

[6] G. Cong, C. S.Jensen, and D. Wu. Efficient retrieval of the top-k most relevant spatial web objects. In *PVLDB*, pages 337–348, 2009.

[7] M. Efron and G. Golovchinsky. Estimation methods for ranking recent information. In *SIGIR*, pages 495–504. ACM, 2011.

[8] P. Haghani, S. Michel, and K. Aberer. Evaluating top-k queries over incomplete data streams. In *CIKM*, pages 877–886, 2009.

[9] P. Haghani, S. Michel, and K. Aberer. The gist of everything new: Personalized top-k processing over web 2.0 streams. In *CIKM*, pages 489–498, 2010.

[10] I.D.Felipe, V.Hristidis, and N.Rishe. Keyword search on spatial databases. In *ICDE*, pages 656–665, 2008.

[11] K. V. R. Kanth, S. Ravada, and D. Abugov. Quadtree and r-tree indexes in oracle spatial: a comparison using GIS data. In *SIGMOD*, pages 546–557, 2002.

[12] F. Korn and S. Muthukrishnan. Influence sets based on reverse nearest neighbor queries. In *SIGMOD*, pages 201–212, 2000.

[13] G. Li, Y. Wang, T. Wang, and J. Feng. Location-aware publish/subscribe. In *KDD*, pages 802–810, 2013.

[14] X. Li and W. B. Croft. Time-based language models. In *CIKM*, pages 469–475. ACM, 2003.

[15] Z. Li, K. C. K. Lee, B. Zheng, W.-C. Lee, D. L. Lee, and X. Wang. Ir-tree: An efficient index for geographic document search. *IEEE Trans. Knowl. Data Eng.*, 23(4):585–599, 2011.

[16] H. Liang, Y. Xu, D. Tjondronegoro, and P. Christen. Time-aware topic recommendation based on micro-blogs. In *CIKM*, pages 1657–1661, 2012.

[17] J. Lu, Y. Lu, and G. Cong. Reverse spatial and textual k nearest neighbor search. In *SIGMOD*, pages 349–360, 2011.

[18] A. Magdy, M. F. Mokbel, S. Elnikety, S. Nath, and Y. He. Mercury: A memory-constrained spatio-temporal real-time search on microblogs. In *ICDE*, pages 172–183, 2014.

[19] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008.

[20] K. Mouratidis, M. Hadjieleftheriou, and D. Papadias. Conceptual partitioning: An efficient method for continuous nearest neighbor monitoring. In *SIGMOD*, pages 634–645, 2005.

[21] K. Pripužić, I. P. Žarko, and K. Aberer. Top-k/w publish/subscribe: Finding k most relevant publications in sliding time window w. In *DEBS*, pages 127–138, 2008.

[22] J. B. Rocha-Junior, O. Gkorgkas, S. Jonassen, and K. Nørvåg. Efficient processing of top-k spatial keyword queries. In *SSTD*, pages 205–222, 2011.

[23] J. Sankaranarayanan, H. Samet, B. E. Teitler, M. D. Lieberman, and J. Sperling. Twitterstand: news in tweets. In *SIGSPATIAL*, pages 42–51, 2009.

[24] A. Shraer, M. Gurevich, M. Fontoura, and V. Josifovski. Top-k publish-subscribe for social annotation of news. *PVLDB*, 6(6):385–396, 2013.

[25] Y. Tao, D. Papadias, and X. Lian. Reverse knn search in arbitrary dimensionality. In *VLDB*, pages 744–755, 2004.

[26] X. Wang, Y. Zhang, W. Zhang, X. Lin, and W. Wang. Ap-tree: Efficiently support continuous spatial-keyword queries over stream. In *ICDE*, 2015.

[27] X. Xiong, M. F. Mokbel, and W. G. Aref. Sea-cnn: Scalable processing of continuous k-nearest neighbor queries in spatio-temporal databases. In *ICDE*, pages 643–654, 2005.

[28] X. Yu, K. Q. Pu, and N. Koudas. Monitoring k-nearest neighbor queries over moving objects. In *ICDE*, pages 631–642, 2005.

[29] C. Zhang, Y. Zhang, W. Zhang, and X. Lin. Inverted linear quadtree: Efficient top k spatial keyword search. In *ICDE*, pages 901–912, 2013.

[30] D. Zhang, K.-L. Tan, and A. K. H. Tung. Scalable top-k spatial keyword search. In *EDBT*, pages 359–370, 2013.