World Wide Web (2023) 26:889–903
https://doi.org/10.1007/s11280-022-01062-x

Continuous spatial keyword query processing over geo-textual data streams

Hongwei Liu[1] · Yongjiao Sun[1] · Guoren Wang[2]

Received: 13 April 2022 / Accepted: 27 April 2022 /
Published online: 11 May 2022

Abstract
Real-time processing of spatial keyword queries has been playing an indispensable role in location-based services. In this light, we propose and study a novel problem of processing continuous spatial keyword queries over geo-textual data streams. We define a new location-based continuously query that enable users to define personalized spatial requirement and textual requirement. Each query continuously feeds users with geo-textual objects that satisfy both spatial and textual requirements set by the query. To process massive-scale continuous spatial keyword queries efficiently, we develop a Continuous Spatial Keyword Query Matching (CSKQM) framework that takes a stream of queries as input and applies hierarchical dynamic grid cells to index each batch of queries. We also propose effective index update algorithm and efficient geo-textual object matching algorithm to process massive-scale continuous spatial keyword queries simultaneously over a stream of geo-textual objects. We conduct comprehensive experimental study on two real datasets to verify the performance of the CSKQM framework.

Keywords Spatial · Keyword · Geo-textual · Stream

1 Introduction

The continued proliferation of Location-based Services (LBS) enables web users and mobile users to publish massive-scale geo-textual objects. Each geo- textual object consists of both location information and text information. In particular, location information can be defined as a geographical coordinate with latitude and longitude (e.g., 39° 31′ 26″ N, 116° 54′ 33″ E), or a semantic location (e.g., Peking University, Haidian, Beijing, China).

✉ Hongwei Liu
ipv65g@163.com

✉ Yongjiao Sun
sunyongjiao@mail.neu.edu.cn

Guoren Wang
wanggrbit@126.com

1 College of Computer Science and Engineering, Northeastern University, Shenyang, China

2 College of Computer Science and Engineering, Beijing Institute of Technology, Beijing, China

Text information can be a plain text document, a set of keywords, or a combination of keywords and values. Additionally, a geo-textual object may contain temporal information such as timestamp and time duration. Geo-textual data has been playing an indispensable role in our modern daily lives. It is ubiquitous in a variety of popular location-based social media and online map services, including but not limited to, geo-tagged microblogging posts (e.g., geo-tweets from Twitter and geo-tagged posts from Weibo), Points of Interest (e.g., Coffee shop in Google Maps), and local news articles. The information from Geo-textual objects may cover a broad range of topics. For example, microblogging posts often offer the quickest first-hand reports of bursty events [1], and geo-tagged documents may be an early indicator of local trending news [2]. As such, it is of great importance to enable web users and mobile users to be updated with most recent geo-textual objects in a continuous fashion.

In this paper, we study the problem of processing Continuous Spatial Keyword (CSK) queries over a stream of geo-textual objects. Specifically, each CSK query is defined by a region of any shapes and a set of keywords connected by AND, OR, or NOT semantics. A CSK query continuously receives geo- textual objects from the input data stream that meet the spatial constraint and textual constraint. In particular, spatial constraint is defined by the spatial region, and textual constraint is defined by the Boolean keyword expression.

Efficient processing of CSK queries has the following technical challenges. First, the number of CSK queries can be very large, it is important to develop an effective scheme to processing massive-scale CSK queries efficiently. Second, each CSK query is required to be processed in a real-time fashion. When a new geo-textual object from data streams arrives, the CSK queries whose constraints can be satisfied by the new object need to be updated instantly. Third, each CSK query may have its unique spatial region and keyword set. Note that the query spatial region can be of any shapes, including but not limited to circle, rectangle, triangle, star, etc. A straightforward method works as follows. Each time when a new geo-textual object o arrives, we calculate whether the spatial and textual information of o meets the spatial and textual constraints, respectively, of each CSK query. If the spatial and textual information of o meets the constraints of query q, we deliver o to q as the result. This method is very time consuming because each time a new object arrives, we need to evaluate whether o matches each CSK query. In real-life scenarios, the number of CSK queries can be very large, which can be million scale or even ten-million scale. At the same time, the geo-textual objects from data streams are arriving at a high rate. As such, we need to evaluate each CSK query against each new object. Hence, it is computationally expensive to apply the straightforward method.

In this light, we propose a novel continuous spatial keyword query matching (CSKQM) framework to process a large number of CSK queries effectively over a stream of geo-textual objects. Specifically, we regard both geo-textual objects and CSK queries as data streams and build CSK query index in an incremental manner. The CSK query index is a hierarchical gird indexing structure that recursively partitions the underlying space into $n \times n$ cells. In particular, we propose a cost model to determine the value of n based on expected computation cost. For each incoming CSK query q, we iteratively find a set of non-overlapping cells from different layers that fully cover the spatial region of q. Next, we generate a “posting” of q, denoting the keywords of q, and store the posting under each cell associated with the spatial region of q. We use inverted file to index the keyword information of CSK queries. Note that each cell maintains its own inverted file, indexing the CSK queries of which spatial regions overlap with the cell. When a new geo-textual object o arrives, we visit all cells from different layers that cover the location of o. For each visited cell, we visit its corresponding inverted file and retrieve the postings. Each posting corresponds to a CSK query. If the posting of query q is retrieved, then o is a result of q and we need to deliver o to q.

Our proposed CSKQM framework has the following major advantages.

- *Scalability*: Our CSKQM framework is capable of handling millions of CSK queries simultaneously because our proposed CSK query index is capable of indexing massive-scale CSK queries in an effective manner.
- *Efficiency*: When a new geo-textual object arrives, our CSKQM framework is able to process each the indexed CSK query set within user interaction time and the real-time results can be guaranteed given millions of indexed CSK queries
- *Generalization*: Our CSKQM framework allows users to issue CSK queries that have different keywords, different connection semantics, and different shapes of spatial regions.

Although the problem of continuous spatial keyword query processing has been extensively investigated by existing studies, to the best of our knowledge, the proposals of existing studies fail to have the aforementioned three advantages simultaneously. Note that all of the aforementioned three aspects, including salability, efficiency, and generalization, are playing an indispensable role in continuous query processing. With the continued proliferation of location-based social media and various location-based Apps, it is becoming increasingly important to develop a scalable, efficient, and generic continuous spatial keyword query processing mechanism.

- We study a new problem of processing a large number of CSK queries in a real-time manner, where each CSK query consists of a set of query keywords connected by AND, OR, or NOT semantics, and a query region of arbitrary shape, which can either be convex shape, concave shape, or multiple shapes.
- We develop a CSKQM framework with a dedicated query indexing structure to organize massive-scale CSK queries effectively. Based on the indexing structure, we propose an online query matching algorithm that is capable of finding a subset of CSK queries that can include each new object as their results in real-time fashion.
- We conduct extensive experiments by using real-life datasets and the experimental results show that our proposal is able to achieve high efficiency and high scalability.

The remaining of this paper is organized as follows. Section 2 defines the geo-textual object, CSK query, and our problem. Section 3 details our proposed solution. Section 4 presents the experimental studies. Section 5 reviews the related work, and Sect. 6 concludes the results.

2 Problem statement

In this section, we present the definition of geo-textual objects, Continuous Spatial Keyword (CSK) query, and our problem formulation.

Definition 1 geo-textual object A geo-textual object is defined by a tuple $o=(\psi, \rho)$, where $o.\psi$ denotes text information, which can be modeled by a sequence of terms, and $o.\rho$ is a geographical point location with latitude and longitude.

The proposal of this paper is designed based on the scenario that the geo-textual objects are arriving in a streaming manner. For example, it can be tweets with location information from Twitter, geo-tagged photos with descriptions from Instagram, check-ins with text and Point of Interests information from Foursquare, local news, etc.

Definition 2 Continuous Spatial Keyword (CSK) query A CSK query is defined by a triple $q=(w, r, s)$, where $q.w$ is a set of query keywords, $q.r$ is a geographical query region, and s is a semantic connection term, which can be AND, OR, or NOT.

Basically, given a stream of geo-textual objects, a CSK query q is to continuously find targeting geo-textual objects where for each targeting geo-textual object o, its text information ($o.\psi$) satisfies the textual condition set forth by $q.w$ and $q.s$, and spatial information ($o.\rho$) satisfies the spatial condition set forth by $q.r$. As such, we define the concept of "matching". Specifically, if a geo-textual object o satisfies both textual condition and spatial condition set forth by CSK query q, then we say object o matches query q.

Definition 3 object-query matching Given a geo-textual object o and a CSK query q, o matches q iff: (1) $o.\psi$ satisfies $q.w$ and $q.s$, and (2) $o.\rho$ is covered by $q.r$.

In this paper, we study the problem of processing a large number of CSK queries over a stream of geo-textual objects. Here, each CSK query is expected to receive real-time results from the geo-textual data stream.

3 CSK query processing

In this section, we first present the baseline solution to processing a large number of CSK queries over a stream of geo-textual objects, which is named as Grid-based Direct Search (Sect. 3.1). Next, we present the details of our CSKQM framework.

3.1 Grid-based direct search

This subsection introduce our grid-based direct search algorithm to process CSK queries. The high-level idea works as follows. First, we partition the underlying space into $n \times n$ grid cells. For each cell, we store the CSK queries whose spatial regions have overlapping areas with the cell. When a new geo- textual object on arrives, we evaluate each CSK queries indexed under the cell that covers the location of the new object. If on matches an indexed query q, we return on as a result of q. Next, we details the query index update algorithm and the object processing algorithm respectively.

3.1.1 Query index update algorithm

Algorithm 1: GridIndexUpdate

Algorithm 1: GridIndexUpdate

```
   Data: CSK query set Q, Grid resolution n
   Result: Grid query index 𝒢
1  Initialize 𝒢 as n × n grid cells;
2  for each q ∈ Q do
3      for each cell c in 𝒢 do
4          if q.r overlaps with c.r then
5              Initialize p(q);
6              p(q).w ← q.w;
7              p(q).s ← q.s;
8              c.add(p(q));
9              Update(𝒢.c);
10 return 𝒢;
```

The pseudo code of the grid query index update algorithm is presented by Algorithm 1. The inputs are CSK query set Q and grid resolution n. The output is the grid query index for indexing queries in Q, which is denoted by G. At the beginning, we initialize the grid index G by $n \times n$ grid cells (Line 1). Next, for each CSK query q in Q, we find a subset of cells that overlap the spatial area of $q.r$ (Lines 3–4). If there exists overlapping area between the spatial area of cell c (i.e., $c.r$) and the spatial region of q (i.e., $q.r$), we index q under c (Lines 4–9). Specifically, we generate a posting for q, which is denoted by $p(q)$. We set the query keyword information of $p(q)$, denoted by $p(q).w$, to be $q.w$, and set the connection semantic of $p(q)$, denoted by $p(q).s$, to be $q.s$ (Lines 6–7). Next, we add the posting of q to c and update the grid query index (Lines 8–9). Finally, we return G as the result (Line 10).

3.1.2 Object processing algorithm

Algorithm 2: ObjectProcessing

Algorithm 2: ObjectProcessing

```
Data: New object o, Query set Q, Grid query index 𝒢
Result: Matching query set R
c ← cell that covers o.ρ in 𝒢;
for each posting p indexed under c do
    q_i ← p.entry;
    if q_i.r covers o.ρ then
        T ← Predicate(q_i.w,q_i.s);
        if o.ψ matches T then
            R.add(q_i);
return R;
```

The pseudo code of the geo-textual object processing algorithm is presented by Algorithm 2. The inputs are the new object o from the geo-textual data stream, existing CSK query set Q, and the grid query index G. The output is the subset of query set $R \subseteq Q$ such that each query $q \in R$ can regard the new object o as one of its result. At the beginning, we locate the cell c in G that covers the location of the new object o (Line 1). Next, we evaluate each posting p indexed under c (Lines 2–7). Specifically, for each posting p we retrieve the corresponding query qi (Line 3). If the spatial region of qi covers the location of o, we proceed to check if qi can textually match o (Lines 4–7). Here, we first generate the text predicate of qi based on $qi.w$ and $qi.s$, which is denoted by T (Line 5). Then we check if $o.\psi$ matches T (Line 6). If so, we add qi into the matching query set R (Line 7). Finally, we return R as the result (Line 8). Note that we need to deliver o to each query in R.

3.2 Continuous spatial keyword query matching framework

The Grid-based Direct Search has the following limitations. First, it is difficult to set an appropriate grid resolution as the spatial region of CSK queries can be varied. Although a higher grid resolution may improve the efficiency of object processing, it may have negative effect on index update because we need more cells to index each query. In contrast, if we set a lower grid resolution, we may save space cost and time cost of index update while decreasing the efficiency of object processing. As such, it is impossible to set a resolution that is feasible to all queries. To address this challenge, we develop a hierarchical grid query indexing structure that is capable of using dynamic grid resolution to index queries based on their spatial locations and shapes.

3.2.1 Hierarchical grid query index

Hierarchical grid query index uses different grid granularity to index the spatial information of CSK q ueries. For each grid cell, we const ruct an inver ted file to index the text ual infor mation of CSK queries whose query regions intersect with the cell. In particular, when a new query arrives, we do not index it immediately. Instead, we temporarily store it in a buffer. When the number of queries reaches the buffer size limit, we perform group query partitioning and find an global optimal partitioning scheme to index the group of queries in the buffer.

Figure 1 illustrates a toy example of our query partitioning scheme to the hierarchical grid query index. Let q1, q2,…, $q6$ be six CSK queries and $q1.r$, $q_2.r$,…, $q_6.r$ be their corresponding query regions, respectively. Let c_1, c_2,…, c9 be nine representative grid cells from different layers. Each CSK query is indexed under a set of grid cells that altogether cover its spatial region. Assume that the current structure of the hierarchical grid query index is illustrated by Figure 1 where the black square denotes the underlying space and the blue segments denote the partitioning of the grid cells. We see that $q1$ is indexed by $c1$ because $q1.r$ intersects with $c1$ only, $q2$ is indexed by both $c1$ and $c2$ since $q2.r$ intersects with both $c1$ and $c2$. Likewise, we see that $q3$ is indexed by $c5$, $c6$, $c7$, $c8$, and $c9$, $q6$ is indexed by $c3$ and $c4$, and $q5$ and $q6$ are indexed by the cells in light red color.

Recall that for each cell, we maintain an inverted file to index the textual information of CSK queries. We proceed to present how to index the textual information of CSK queries. According to the definition of the CSK query, we need to support AND, OR, and NOT semantics. However, traditional inverted file is designed for plain text document, which is inapplicable to indexing the aforementioned query predicates. For the purpose, we design a novel query inverted file dedicated for the textual information of CSK queries. We design three schemes to handle query keywords connected by AND, OR, and NOT semantics, respectively.

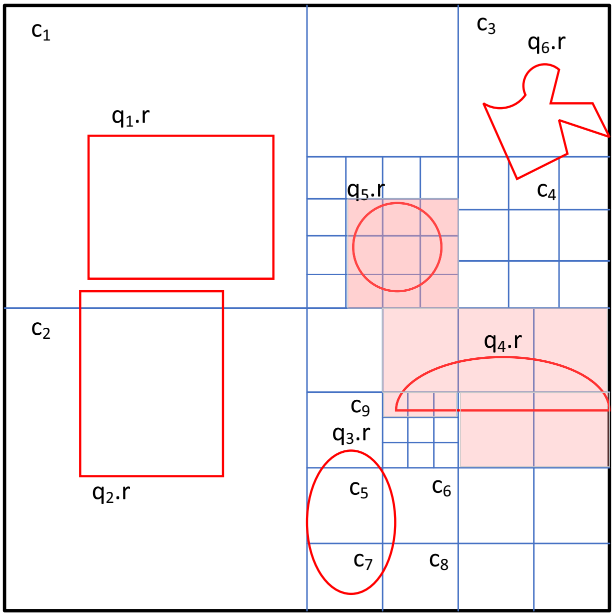

Figure 1 Hierarchical grid query index

Specifically, given a CSK query q, if $q.s$ is OR, we create $|q.w|$ postings and each posting is associated to an individual query keyword. If $q.s$ is AND, we only create one posting and the posting is associated the query keyword with the least frequency. If $q.s$ is NOT, we do not let q be indexed by inverted file. Instead, we store q separately to a list exclusively designed for queries that have NOT semantic.

Algorithm 3 presents the pseudo code of our hierarchical grid index update scheme. The inputs are (1) a batch of CSK queries B in the buffer; (2) the current cell c; (3) step resolution threshold m, denoting that the cell partitioning between two consecutive layers cannot exceed $m \times m$; (4) coverage ratio threshold θ, denoting the termination condition regarding the ratio of the query region size and the sum of cell sizes that index the query. The output is the updated index structure that takes queries of B into consideration. At the beginning, we set a initial value of resolution to be 2×2 (Line 1). For each resolution from 2×2 to $m \times m$, we perform the following steps. To be specific, we first initialize the sub-index rooted at cell c to be $k \times k$ grid cells (Line 3).

Then we initialize $C\ Ravg$ to be 0 (Line 4). Note that $C\ Ravg$ represents the average coverage ratio of queries in B. For each query q in B, we first initialize Cq, representing the set of cells that intersect with $q.r$, as an empty set (Line 6). For each cell ci in Gc, we check if ci intersects with $q.r$. If so, we add ci to set Cq (Lines 8–9). Next, we calculate the coverage ratio of q, which is denoted by $C\ R(q)$. Note that $C\ R(q)$ is computed by dividing the area of $q.r$ to the sum of areas of Cq (Line 10). After evaluating all queries in B, we calculate the average coverage ratio of queries in B (Line 12). If the average coverage ratio is no less than the pre-defined coverage threshold θ, we stop evaluating the partitioning with finer resolution (Lines 13–14). Otherwise, we proceed with finer resolution by increasing k by 1. If k reaches m or the average coverage ratio is no less than the pre-defined coverage threshold θ, we stop the resolution evaluation.

Next, we store the posting of each query in B to the corresponding cells. Finally, we update G with Gc and return the updated G as the updated index.

Algorithm 3: HierarchicalIndexUpdate

Algorithm 3: HierarchicalIndexUpdate

Data: CSK query batch B, Current cell c, Step resolution threshold m, Coverage ratio threshold θ

Result: Update of hierarchical grid query index $\mathcal{G}$

$k \leftarrow 2$;
do
    Initialize $\mathcal{G}_c$ as $k \times k$ grid cells;
    $CR_{avg} \leftarrow 0$;
    for *each* $q \in B$ **do**
        $C_q \leftarrow \emptyset$;
        for *each cell* $c_i \in \mathcal{G}_c$ **do**
            if c_i *intersects with* $q.r$ **then**
                C_q.add(c_i);
        $CR(q) \leftarrow \frac{Area(q.r)}{Area(C_q)}$;
        $CR_{avg} \leftarrow CR_{avg} + CR(q)$;
    $CR_{avg} \leftarrow \frac{CR_{avg}}{|B|}$;
    if $CR_{avg} \geq \theta$ **then**
        break;
    $k \leftarrow k + 1$;
while $k \leq m$;
for *each cell* c_i *in* $\mathcal{G}_c$ **do**
    for *each* $q \in B$ **do**
        if $q.r$ *overlaps with* $c_i.r$ **then**
            Generate the posting of q;
            Update inverted file associated with c_i;
Update $\mathcal{G}$ with $\mathcal{G}_c$
return $\mathcal{G}$;

4 Experimental study

In this section, we conduct extensive experiments to evaluate the performance of our proposal.

4.1 Baseline

A straightforward method is presented in Sect. 3.1. It basically uses a grid- based indexing structure to organize CSK queries. We use it as our baseline. Note that we tune the grid resolution based on the size of query regions.

4.2 Datasets and generation of CSK queries

We use two real-world datasets in our experiments: FQ and TE. FQ is a real-world dataset collected from Foursquare, which contains 2 million worldwide check-ins where each check-in has both geographical location and text document. Dataset TE is a real-world dataset as well, which contains 10 million geo-tagged tweets. Each geo-tagged tweet has both location and text information.

The CSK queries are generated as follows. For each geo-textual object, which is check-in or geo-tagged tweet in FQ and TE, respectively, we randomly select a number of keywords from the object keywords. As for the query spatial region, we generate three types of regions: circle, square, and hexagon. We regard the location of each randomly selected geo-textual object as the center of query spatial region, and let the size of the region be of pre-defined size.

4.3 Experimental settings

Our parameter settings used in the experiments are presented in Table 1. Note that we use <u>GDS</u> to denote our baseline method and use <u>CSKQM</u> to denote the method proposed in this paper.

4.4 Experimental result

We proceed to present our experimental results.

4.4.1 Effect of the number of query keywords

This set of experiments investigates the effect of the number of CSK query keywords regarding both methods. Figure 2 shows the results on FQ and TE datasets respectively. We could see that the runtime of geo-textual object processing for both methods exhibit an increasing trend when we increase the number of query keywords. The reason is that when the number of query keywords increases, the index sizes of both GDS and CSKQM become larger. As such, it may take more time to retrieve the spatially relevant and textually relevant queries when a new object arrives.

At the same time, we find that CSKQM performs consistently better than GDS for approximately 4X to 6X regarding the efficiency of object processing. Such significant

Table 1 Parameter settings

Parameter	Setting	Default
Number of query keywords	1 – 5	3
Grid resolution	1 km – 50 km	On the basis of scenarios
Query region size	1km^2 – 400km^2	Random
Step resolution threshold m	3 – 6	4
Coverage ratio threshold θ	0.2 – 0.8	0.6
Number of CSK queries	On the basis of scenarios	FQ:1 M TE 5 M

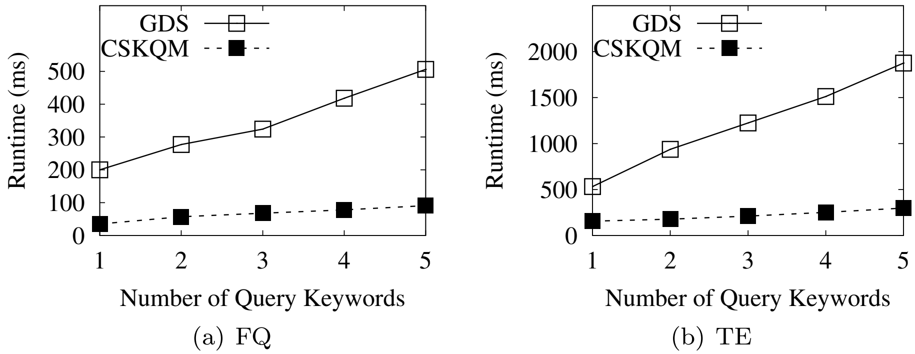

Figure 2 Effect of the number of query keywords

performance improvement is resulted from the dynamic grid granularity provided by the hierarchical partitioning scheme.

4.4.2 Effect of query region size

Figure 3 demonstrates the results on FQ and TE respectively when we vary the size of query spatial regions. We see that GDS performs worse when we enlarge the query region size. The reason can be explained as follows. When the query region size becomes larger, if the grid resolution is constant, we need to use more cells to index each CSK query. As a result, the index size may become larger. When a new object arrives, we are expected to compare the new objects with more postings. Hence, the runtime of object processing will be increased. In particular, we also find that when we increase the region size from 1 km^2 to 100 km^2, the runtime of GDS only exhibits a slight increasing trend. In contrast, when we proceed to increase the region size from 100 km^2 to 400 km^2, the runtime of GDS only exhibits a sharp increasing trend. Such contrast can be explained by the fact that when the region size is smaller than the cell size, GDS only needs to use a small number of cells to index the query. However, when the query region size is significantly larger than the cell size, the number of cells required to index a query may be proportional to the size of the query region.

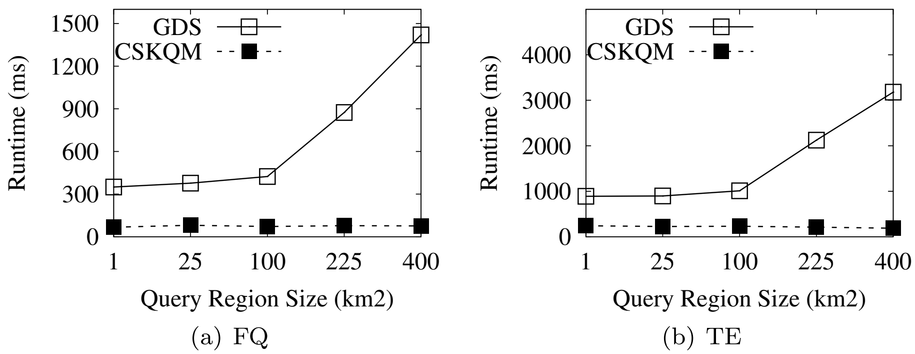

Figure 3 Effect of query region size

Compared to GDS, our proposed CSKQM does not exhibit similar performance trend. In contrast, the object processing time of CSKQM is relatively consistent as we vary the query region size. The reason is that CSKQM is capable of using different grid resolutions to index CSK queries with different regwn sizes.

4.4.3 Effect of step resolution threshold

Figure 4 presents the performance result of CSKQM when we vary the step resolution threshold *m* from 3 to 6. We see that the time cost of object processing decreases when we increase the step resolution threshold. The reason is that when we increase the step resolution threshold, it is more likely for a query to be partitioned and represented by a set of fine-grained cells. As such, queries are indexed in a more precise way, which in turn may enhance the efficiency of object processing.

4.4.4 Effect of coverage ratio threshold

Figure 4 presents the performance result of CSKQM when we vary the coverage ratio threshold θ from 0.2 to 0.8. We see that the performance of CSKQM becomes better on both datasets when we increase the step resolution threshold. The reason is that when we incease the step resolution threshold, it denotes that we impose a more rigorous requirement in query region partitioning. As such, the spatial region of each query may be represented more precisely (Figure 5).

4.4.5 Effect of the number of indexed queries

Finally, we evaluate the object processing performance when we increase the number of indexed queries. From Figure 6 we see that both methods performs worse when we increase the number of indexed queries. The reason is quite straightforward. When the number of indexed queries mounts up, the postings list maintained by each cell is expected to be longer. We also find that CSKQM performs consistently better than GDS for at least 5X.

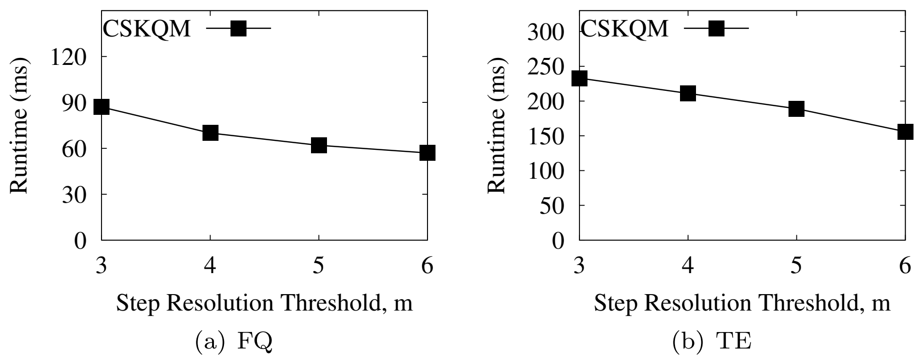

Figure 4 Effect of the step resolution threshold, *m*

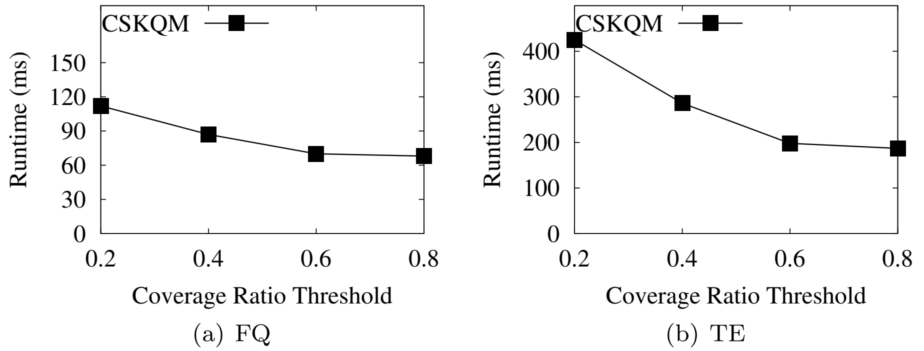

Figure 5 Effect of coverage ratio threshold, θ

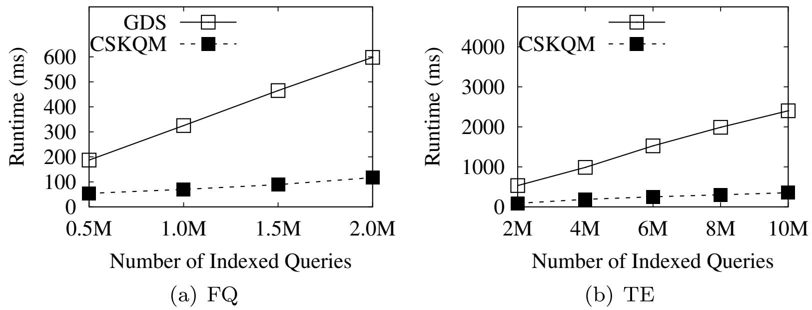

Figure 6 Effect of the number of indexed queries

5 Related work

In this section, we investigate relevant related studies regarding spatial keyword search and location-based continuous query processing.

5.1 Spatial keyword search

The problem of spatial keyword search can be defined as processing spatial keyword queries over a collection of geo-textual objects. A spatial keyword query consists of both spatial query component and textual query component. A spatial query component can be a region, a distance threshold, and spatial proximity. A textual query component can be a Boolean textual predicate and a set of keywords. Based on the spatial and textual information, spatial keyword queries can be classified as Boolean Range query, Boolean *k*-NN query, and Top-*k* *k*-NN query. Efficient processing of spatial keyword queries has been extensively studied by existing works [3–13]. Some surveys and experimental studies regarding spatial keyword search techniques can be found as well [14–16]. Recently, some studies focus on pattern mining over sequential geo-textual data, which is named as semantic trajectory data [17–19]. These studies have wide and practical applications and they are on the basis of spatial keyword search techniques.

However, the aforementioned studies regard geo-textual objects as a collection of static data or a group of items with low-frequent updates. Moreover, their queries are one-time queries, which means that each query is only responsible for the results at the snapshot when the query is issued. In contrast, our problem is to handle a stream of geo-textual objects and the CSK queries defined in this paper may continuously receive up-to-date results over time. No sensible way exists to use the methods proposed by aforementioned studies for our problem.

5.2 Location and content based continuous query processing

Existing studies model the problem of continuous content-based query processing as the problem of publish/subscribe. A host of studies that target the problem of developing efficient publish/subscribe algorithms [1, 20–24]. A traditional publish/subscribe framework consists of publisher component and subscriber component. Specifically, publisher component can be regarded as a stream of items while subscriber component can be regarded as a set of subscribers where each subscriber continuously receives targeting items from the publisher side. Over the last decade, some studies enable subscribers to defined their location-based requirements, which is called location-based publish/subscribe [25–33].

However, the aforementioned publish/subscribe framework has the following limitations. First, it has specific requirements towards the shape of query spatial region. Second, it only supports limited textual connection semantics. It is an open problem regarding how to support all of the three major textual connection semantics, namely AND, OR, and NOT semantics.

6 Conclusions

We consider the problem of processing a large number of CSK queries over a stream of geo-textual objects. We define a new type of location-based continuous query that supports arbitrary shape of query spatial regions and supports all of the three major textual connection semantics, including AND, OR, and NOT semantics. To process a large number of CSK queries efficiently, we develop a CSKQM framework that takes a stream of CSK queries as input and use hierarchical dynamic grid cells to index each batch of CSK queries. We also propose effective index update algorithm and efficient geo-textual object matching algorithm to process massive-scale CSK queries simultaneously over a stream of geo-textual objects. The experimental results on two real-world datasets show that our proposal, CSKQM framework, is capable of achieving a runtime reduction of 70%-85% compared with baseline.

Acknowledgements The work is supported by the National Natural Science Foundation of China (Grant No. 61972077), LiaoNing Revitalization Talents Program (Grant No. XLYC2007079), the Science and Technology Plan Project of Shen Fu Reform and Innovation demonstration Zone in 2021 (Big Data Deep Analysis Platform for New Energy Vehicles).

Author contribution Hongwei Liu: Algorithm design and development, and paper writing

Yongjiao Sun: Experimental study

Guoren Wang: Algorithm design, and paper proofreading

All authors reviewed the manuscript.

Funding The work is supported by the National Natural Science Foundation of China (Grant No. 61972077), LiaoNing Revitalization Talents Program (Grant No. XLYC2007079), the Science and Technology Plan Project of Shen Fu Reform and Innovation demonstration Zone in 2021 (Big Data Deep Analysis Platform for New Energy Vehicles).

Data Availability Not applicable.

Declarations

Ethical approval and consent to participate Not applicable.

Human and animal ethics Not applicable.

Consent for publication Not applicable.

Competing interests The authors declare that they have no competing interests.

References

1. Shraer, A., Gurevich, M., Fontoura, M., Josifovski, V.: Top-k publish-subscribe for social annotation of news. PVLDB **6**(6), 385–396 (2013)
2. Chen, L.,Cong, G., Cao, X., Tan, K.-L.: Temporal spatial-keyword top-k publish/subscribe. In ICDE, pp 255–266. (2015)
3. Felipe, I.D., Hristidis, V., Rishe, N.: Keyword search on spatial databases. In ICDE, pp. 656–665. (2008)
4. Cong, G., Jensen, C.S., Wu, D.: Efficient retrieval of the top-k most relevant spatial web objects. In PVLDB, pp. 337–348. (2009)
5. Rocha-Junior, J.B., Gkorgkas, O., Jonassen S., Nørv˚ag, K.: Efficient processing of top-k spatial keyword queries. In SSTD, pp. 205–222. (2011)
6. Zhang,D., Tan, K.-L., Tung, A.K.H.: Scalable top-k spatial keyword search. In EDBT, pp. 359–370. (2013)
7. Zhang,C., Zhang, Y., Zhang, W., Lin, X.: Inverted linear quadtree: Efficient top k spatial keyword search. In ICDE, pp. 901–912. (2013)
8. Wu, D., Yiu, M.L., Jensen, C.S., Cong, G.: Efficient continuously moving top-k spatial keyword query processing. In ICDE, pp. 541–552. (2011)
9. Yang, C., Chen, L., Shang, S., Zhu, F., Liu, L., Shao, L.: Toward efficient navigation of massive-scale geo-textual streams. In IJCAI, pp. 4838–4845. (2019)
10. Li, M., Chen, L., Cong, G., Gu, Y., Yu, G.: Efficient processing of location-aware group preference queries. In CIKM, pp. 559–568. ACM (2016)
11. Kalamatianos, G., Fakas, G.J., Mamoulis, N.: Proportionality in spatial keyword search. In SIGMOD ’21: International Conference on Management of Data, Virtual Event, China, June 20–25, 2021, pp. 885–897. ACM (2021)
12. Jiajie, Xu., Sun, J., Zhou, R., Liu, C., Yin, L.: CISK: an interactive framework for conceptual inference based spatial keyword query. Neurocomputing **428**, 368–375 (2021)
13. Chen, X., Jiajie, Xu., Zhou, R., Zhao, P., Liu, C., Fang, J., Zhao, L.: S^2r-tree: a pivot-based indexing structure for semantic-aware spatial keyword search. GeoInformatica **24**(1), 3–25 (2020)
14. Chen, L., Cong, G., Jensen, C.S., Wu, D.: Spatial keyword query processing: an experimental evaluation. In PVLDB, pp. 217–228. (2013)
15. Chen, L., Shang, S., Yang, C., Li, J.: Spatial keyword search: a survey. GeoInformatica **24**(1), 85–106 (2020)
16. Chen, Z., Chen, L., Cong, G., Jensen, C.S.: Location- and keyword- based querying of geo-textual data: a survey. VLDB J. **30**(4), 603–640 (2021)
17. Zhang, C., Han, J., Shou, L., Jiajun, Lu., La Porta, T.: Splitter: Mining fine-grained sequential patterns in semantic trajectories. Proc VLDB Endow **7**(9), 769–780 (2014)
18. Renhe, J., Jing, Z., Tingting, D., Yoshiharu, I., Chuan, X., Yuya, S.: A density-based approach for mining movement patterns from semantic trajectories. In TENCON 2015–2015 IEEE Region 10 Conference, pp. 1–6. IEEE (2015)

19. Shan, Z., Sun, W., Zheng, B.: Extract human mobility patterns powered by city semantic diagram. IEEE Transactions on Knowledge and Data Engineering, (2020)
20. Pripuˇzi´c, K., Zarko, I.P., Aberer, K.: Top-k/w publish/subscribe: Finding k most relevant publications in sliding time window w. DEBS, pp. 127–138. (2008)
21. Haghani, P., Michel, S., Aberer, K.: Evaluating top-k queries over incomplete data streams. In CIKM, pp. 877–886. (2009)
22. Haghani, P., Michel, S., Aberer, K.: The gist of everything new: Personalized top-k processing over web 2.0 streams. In CIKM, pp. 489–498. (2010)
23. Chen, L., Cong, G.: Diversity-aware top-k publish/subscribe for text stream. In SIGMOD, p. 347–362. (2015)
24. Machanavajjhala, A., Vee, E., Garofalakis, M., Shanmugasundaram, J.: Scalable ranked publish/subscribe. PVLDB **1**(1), 451–462 (2008)
25. Li,G., Wang, Y., Wang, T., Feng, J.: Location-aware publish/subscribe. In KDD, pp. 802–810. (2013)
26. Chen, L., Cong, G., Cao, X.: An efficient query indexing mechanism for filtering geo-textual data. In SIGMOD, pp. 749–760. (2013)
27. Chen, L., Cui, Y., Cong, G., Cao, X.: SOPS: A system for efficient processing of spatial-keyword publish/subscribe. PVLDB **7**(13), 1601–1604 (2014)
28. Chen, L., Shang, S., Jensen, C.S., Jianliang, Xu., Kalnis, P., Yao, B., Shao, L.: Top-k term publish/subscribe for geo-textual data streams. VLDB J **29**(5), 1101–1128 (2020)
29. Chen, L., Shang, S.: Approximate spatio-temporal top-k publish/subscribe. World Wide Web **22**(5), 2153–2175 (2019)
30. Chen, L., Shang, S., Zhang, Z., Cao, X., Jensen, C.S., Kalnis, P.: Location-aware top-k term publish/subscribe. In ICDE, pp. 749–760. IEEE Computer Society (2018)
31. Chen, Z., Cong, G., Zhang, Z., Fu, T.Z.J., Chen, L.: Distributed publish/subscribe query processing on the spatio-textual data stream. In ICDE, pp. 1095–1106. IEEE Computer Society (2017)
32. Wang, X., Zhang, W., Zhang, Y., Lin, X., Huang, Z.: Top-k spatial-keyword publish/subscribe over sliding window. VLDB J. **26**(3), 301–326 (2017)
33. Wang, X., Zhang, Y., Zhang, W., Lin, X., Wang, W.: Ap-tree: efficiently support location-aware publish/subscribe. VLDB J. **24**(6), 823–848 (2015)