World Wide Web (2023) 26:933–947
https://doi.org/10.1007/s11280-022-01063-w

Continuous similarity join over geo-textual data streams

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

Received: 13 April 2022 / Accepted: 28 April 2022 /
Published online: 2 June 2022

Abstract
Geo-textual similarity join is a fundamental operation in spatial databases. With the continued proliferation of location-based social media, geo-textual data is becoming increasingly available over the past decades. Mobile users and location-based service providers may want to receive up-to-date similarity join results of massive-scale geo-textual objects over data streams. In this light, we propose and study a novel problem of Continuous Geo-Textual Similarity Join (CGTS-Join). Specifically, given a collection of geo-textual objects Q and a dynamic set of geo-textual objects P over geo-textual data streams, the problem CGTS-Join is to continuously maintain an up-to-date join result set containing object pairs such that the objects of each pair are similar to each other. For the purpose, we define an effective similarity metric that measures the similarity between two geo-textual objects by taking spatial, textual, and temporal aspects into consideration. Based on the similarity metric, we develop a Hybrid Grid Indexing Structure (HGI) and a tri-filtering framework that is capable of answering the CGTS-Join problem efficiently. We conduct extensive experiments on two real-world datasets to confirm the performance superiority of our proposed method.

Keywords Spatial · Keyword · Geo-textual · Stream

1 Introduction

The continued development of location-based social media enables the omnipresence of geo-textual data. For example, Twitter, a popular micro-blogging service, enables mobile users to publish a short-text post with textual, spatial, and temporal information. Other social media platforms such as Facebook, WeChat, Foursqure, allow mobile users to publish items with textual, spatial, and time information as well. These items can be modeled

✉ 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

as geo-textual objects. Each geo-textual object consists of text document, spatial coordinate, and timestamp. As the number of geo-textual objects has been skyrocketing over the past decades, it is of great importance to enable basic data analytic functionalities regarding massive-scale geo-textual data. In particular, similarity join, as one of the most popular data analytic functionalities, has been extensively investigated by existing studies. Similarity join operation has been playing an important role in spatial data management as it has a broad range of applications, including but not limited to data cleaning, data summarization, ridesharing recommendations, and spatial keyword query result authentication.

However, existing studies regarding similarity join has the following limitations. First, they do not consider the similarity join in geo-textual domain where spatial, textual, and temporal similarities are required to be taken into account. Second, existing approaches are designed on the basis of static scenario. That is, underlying data objects are regarded as a collection of items. However, in many real-life applications the data objects are arriving in a streaming fashion. It is of great importance to enable real-time processing of similarity join for geo-textual data streams.

In this paper, we investigate the problem of similarity join over a stream of geo-textual objects. In particular, given a stream of geo-textual objects P and a set of geo-textual object Q, we study the problem of finding all pairs of geo-textual objects (o_i,o_j) where $o_i \in P$ and $o_j \in Q$ and the similarity between o_i and o_j is no less than a similarity threshold θ. Note that the update frequency of P is much more higher than that of Q. We need to consider the following three aspects when measuring the similarity between two geo-textual objects: (1) spatial proximity; (2) textual relevancy; (3) temporal gap.

Efficient processing of geo-textual similarity join under the aforementioned scenario has the following technical challenges. First, we need to define an effective similarity metric to measure the similarity between two geo-textual objects. The similarity metric should take spatial proximity, textual relevancy, and temporal gap between two objects into consideration. Further, the similarity function is expected to be light-weighted. Second, we the number of geo-textual objects can be very large. It is very often that we need to process million of or even tens of millions of geo-textual objects. As such, our solution is required to be scalable. Third, in real-life applications, geo-textual data streams may have a high arrival rate. Our solution needs to meet the efficiency requirement and return real-time join results over dynamic datasets. A straightforward approach works as follow. Given a collection of query objects Q and a stream of geo-textual objects P, each time when a new object o_n arrives to P, we calculate the similarity between o_n and each query object o_q in Q. If the similarity between o_n and o_q is no less than the similarity threshold θ, we add pair (o_n,o_q) into the join result set. This approach is time consuming because we the number of query objects can be very large. It is difficult to meet both efficiency and scalability requirements.

In this light, we develop a novel three-phase filtering approach, which is named as *tri-filtering*, to process the spatio-temporal similarity join problem in a real-time fashion over geo-textual data streams. To be specific, tri-filtering mechanism indexes one of a geo-textual object set that is updated less frequently and regards the other geo-textual object set as a data stream. Then, we propose a hybrid grid indexing structure, HGI, that effectively combines the spatial information, textual information, and temporal information of geo-textual objects. In particular, HGI partitions first partitions the underlying space into a set of $m \times m$ cells. Each cell maintains an inverted file and a sequence of time slots. Note that the inverted file indexes objects located in the cell on the basis of their terms, and the time slots partition the time space into a set of time interval with equal timespan. Each time slot indexes geo-textual objects whose timestamps are within the slot. When a new geo-textual object arrives, we compute a spatial similarity upper bound between the new object and the objects in each cell. If the spatial similarity upper bound is smaller than the similarity threshold θ, we may prune all objects indexed under the corresponding cell safely. If a cell cannot be pruned based on the spatial similarity upper bound, we proceed to evaluate each time slot in the cell and calculate a spatio-temporal similarity between the new object and the objects in each time slot associated to the cell. If the spatio-temporal similarity upper bound is smaller than the similarity threshold θ, we may prune the objects indexed under the correesponding time slot safely. Otherwise, we proceed to evaluate each posting in the inverted file associated with the time slot of the cell.

Our proposed tri-filtering framework has the following major advantages.

– *Effectiveness*: The tri-filtering framework is able to generate real-time join results for massive-scale geo-textual data. It is capable of handling geo-textual stream with high arrival rate.
– *Scalability*: In real-life applications, the throughput of geo-textual data streams can be very high. Thanks to the multi-layer filtering techniques, the tri-filtering framework is able to process the similarity join problem over a large number of geo-textual data simultaneously.
– *Generalization*: The tri-filtering framework is independent of the similarity metrics. It is applicable to a variety of spatial, temporal, and textual similarity functions.

Even if the problem of geo-textual similarity join has been investigated by existing studies, as far as we have concerned, existing studies on this matter are incapable of addressing the aforementioned three factors at the same time. It is worthy of noting that all of the aforementioned three factors, namely effectiveness, scalability, and generalization, are playing important roles in geo-textual similarity join. As the location-based services and social media platforms are becoming increasingly popular, it is imperative to develop an effective, scalable, and generic real-time mechanism to answer the geo-textual similarity join problem.

To sum up, we have made the following contributions.

– We study a new problem of real-time geo-textual similarity join, which has a broad range of applications, such as geo-textual data cleaning, geo-textual data analytics, and geo-textual data visualizations.
– We develop a tri-filtering framework with a dedicated geo-textual object indexing structure to organize massive-scale geo-textual objects effectively. Based on the indexing structure, we propose a couple of filtering techniques and an online geo-textual object matching algorithm. With the aforementioned techniques, tri-filtering is capable of generating geo-textual similarity join results in real-time fashion.
– We conduct extensive experiments over real-world datasets. Our experimental results show that tri-filtering is able to achieve high effectiveness and high scalability.

The remaining parts of this paper are organized as follow. Section 2 presents the definition of geo-textual object and our real-time geo-textual similarity join problem. Section 3 introduces our proposed geo-textual similarity metric. Section 4 details our proposed solution. Section 5 conducts the experimental studies. Section 6 presents the related studies. Section 7 concludes this paper.

2 Problem statement

This section presents the definition of geo-textual objects, geo-textual similarity join, and our problem formulation.

Definition 1 **Geo-textual Object.** A geo-textual object can be defined as a triple $o = \langle \psi,\rho,t \rangle$, where $o.\psi$ denotes text description, $o.\rho$ denotes a geographical location (latitude and longitude), and $o.t$ denotes a timestamp.

Next, we present our definition of continuous geo-textual similarity join problem.

Definition 2 **Continuous Geo-Textual Similarity Join (CGTS-Join).** Given a collection of geo-textual objects Q, a stream of geo-textual objects P, and a similarity threshold θ, the CGTS-Join problem aims to maintain a set of geo-textual object pairs $\mathcal{S}$ where each object pair $(o_i, o_j) \in \mathcal{S}$ satisfies the following conditions:

(1) $o_i \in P$ and $o_j \in Q$;
(2) The similarity between o_i and o_j, denoted by $Sim(o_i,o_j)$, is no less than θ.

Basically, we regard object set Q as a static set while regard set P as a dynamic set. Each time when a new object o_n is inserted into P, we need to update the result set $\mathcal{S}$ with new similar pairs that contain o_n.

3 Geo-textual similarity metric

In this section, we present our geo-textual similarity metric. Note that when we measure the similarity between two geo-textual objects, we need to take the following three aspects into consideration: (1) spatial proximity; (2) textual relevancy; (3) time gap. At the same time, based on our application scenario, the proposed similarity metric is supposed to be calculated without much difficulty. That is, the computation of the similarity between two geo-textual objects needs to have relatively low time complexity. For the purpose, we devise the following similarity function the measure the similarity between two objects o_i and o_j.

$$\begin{aligned} Sim(o_i, o_j) = \ & \alpha \times Sim_{spatial}(o_i.\rho, o.j.\rho) + \beta \times Sim_{textual}(o_i.\psi, o_j.\psi) \\ & + \gamma \times Sim_{temporal}(o_i.t, o_j.t), \end{aligned} \tag{1}$$

where $Sim_{spatial}(o_i.\rho,o.j.\rho)$ denotes the spatial proximity between o_i and o_j, $Sim_{textual}(o_i.\psi,o_j.\psi)$ denotes the textual relevancy between o_i and o_j, $Sim_{temporal}(o_i.t,o_j.t)$ denotes the time gap score between o_i and o_j, and α, β, and γ denote the preference parameters representing the weights of spatial proximity, textual relevancy, and time gap score, respectively. Note that the preference parameters, α, β, and γ, should satisfy the following requirement.

$$\alpha + \beta + \gamma = 1$$

We proceed to introduce how to compute the spatial proximity, textual relevancy, and time gap score, respectively, as follow. We first present how to compute the spatial proximity between o_i and o_j, which is denoted by $Sim_{spatial}(o_i.\rho,o.j.\rho)$ in (2).

$$Sim_{spatial}(o_i.\rho, o.j.\rho) = 1 - \frac{dist(o_i.\rho, o.j.\rho)}{dist_{max}}, \tag{2}$$

where $dist_{max}$ denotes the maximum possible distance in the underlying space. The textual relevancy between o_i and o_j, denoted by $Sim_{textual}(o_i.\psi, o.j.\psi)$, is computed by (3).

$$Sim_{textual}(o_i.\psi, o.j.\psi) = \frac{|o_i.\psi \cap o_j.\psi|}{|o_i.\psi \cup o_j.\psi|}. \tag{3}$$

Note that apart from (3) our framework may support other kinds of text similarity measurement such as cosine similarity and language model. Finally, we present how to compute the time gap score between o_i and o_j, which is defined by (4).

$$Sim_{temporal}(o_i.t, o.j.t) = 1 - \frac{|o_i.t - o_j.t|}{t_{current} - t_{earliest}}, \tag{4}$$

where $t_{current}$ denotes the current timestamp and $t_{earliest}$ denotes the earliest timestamp among the indexed geo-textual objects.

4 Continuous geo-textual similarity join

In this section, we first present the baseline solution to answering the CGTS-Join problem, which is named as Brute Force Matching (BFM) Algorithm (cf. Section 4.1). Next, we present the details of our tri-filtering framework (cf. Section 4.2).

4.1 Brute force matching algorithm

This subsection present the baseline algorithm to answer the CGTS-Join problem. The high-level idea is as follows. Each time when a new geo-textual object o_n arrives, we compute the similarity score between o_n and each indexed object o_i. If the similarity between o_n and o_i is no less than the similarity threshold θ, we generate object pair (o_n,o_i) and add the pair into the join result set $\mathcal{S}$. In this way, $\mathcal{S}$ can be continuously updated as the arrival of new objects.

We proceed to analyze the time complexity of the BFM algorithm. Let P and Q be two sets of geo-textual objects. Assume that P is updated very frequently and Q is a static set. Each time when a new object is inserted into P, we need to compute the similarity between the new object and every object in Q. As such, the time complexity of processing each new object is $O(|\psi|\times|Q|)$, where $|\psi|$ denotes the number of terms per object and $|Q|$ denotes the cardinality of Q. Because we need to process all of the objects in P, the overall time complexity of BFM algorithm can be $O(|\psi|\times|P|\times|Q|)$,

4.2 Tri-filtering framework

The BFM algorithm has the following limitations. First, each time a new object arrives, we need to calculate the similarity between the object and all of the existing objects, which is extremely time consuming. Second, the number of existing geo-textual objects may become larger as time goes by. As such, it is inevitable that the time cost of processing a new object may exhibit an linear-increasing trend, which is inapplicable to the scenario with data streams. To address the aforementioned limitations, we develop a dedicated H ybrid G rid I ndexing structure, HGI, that is able to organize the spatial, textual, and temporal information of geo-textual objects in an effective manner. Based on the indexing structure, we propose pruning techniques based on similarity upper bounds. We also propose efficient object matching algorithms based on the pruning techniques.

The details of HGI will be presented in Section 4.2.1. The group filtering techniques and object matching algorithms will be detailed in Section 4.2.2.

4.2.1 Hybrid grid index

The Hybrid Grid Index, HGI, is designed for organizing geo-textual objects in an effective manner. Geo-textual objects are organized on the basis of *spatial-temporal-textual* hierarchy. Given a collection of geo-textual objects, we first organize the geo-textual objects based on their spatial information. Next, we organize the geo-textual objects based on their temporal information. Finally, objects are indexed based on their textual information. We proceed to introduce how to organize geo-textual objects based on spatial information, temporal information, and textual information, respectively.

To organize the spatial information of geo-textual objects, the HGI applies grid indexing structure. Note that the grid indexing structure partitions the underlying space into $m \times m$ grid cells. For each cell c, we store a subset of geo-textual objects whose locations fall within the spatial range indicated by cell c. Note that m is a system parameter, which is determined based on the spatial distribution of a particular dataset.

To organize the temporal information of geo-textual objects, the HGI uses bucket indexing structure. Specifically, for each grid cell c, we further partition the geo-textual objects indexed under c into b buckets, where b is a system parameter that is determined based on the temporal distribution of the dataset. Each bucket B under cell c is associated with a timespan $[t_a,t_b]$ where t_a denotes the earliest possible timestamp of B and t_b indicates the latest possible timestamp of $c.B$. When a new geo-textual object o_n arrives, if $o_n.\rho$ falls in cell c and $o_n.t$ falls within timespan $[t_a,t_b]$, we let $c.B$ store o_n.

To organize the textual information of geo-textual objects, the HGI uses inverted file for the purpose. In particular, for each bucket B connected to each grid cell c, we maintain its corresponding inverted file that indexes the geo-textual objects whose locations fall in the spatial range of c and whose timestamps fall within timespan $[t_a,t_b]$. The inverted file consists of a set of inverted lists. Each inverted list is associated to a particular keyword. Inverted list is also called postings list. Each postings list consists of a set of postings. Each posting is an entry of a particular geo-textual object. Figure 1 illustrates the structure of inverted file associated to particular buckets. We see that given a grid cell c, the geo-textual objects located within c are further partitioned by four buckets, namely B_1, B_2, B_3, and B_4. Each bucket corresponds to a particular timespan. Note that the timespans of different buckets are mutually exclusive. Each bucket is associated to an inverted file, which is called postings list. The detail of a postings list is illustrated by Figure 2. From Figure 2 we see that bucket B_1 is associated to a postings list. The postings list consists of a number of linked lists. Each linked list corresponds to a keyword, which is denoted by w_1, w_2,..., or w_6. Each linked list consists of postings. Each posting contains an object id and a pointer to the object. If the posting of object o appears in the linked list of keyword w_1, then it means that $o.\psi$ contains w_1.

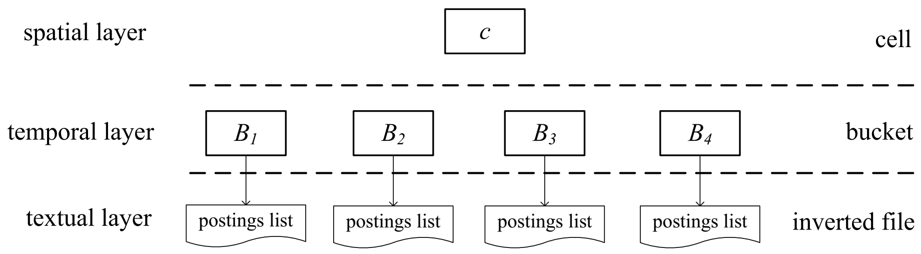


Figure 1 Inverted files associated to a buckets

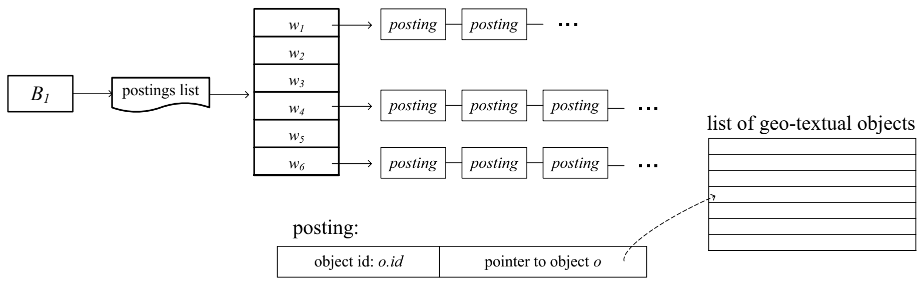


Figure 2 Postings list of inverted file

4.2.2 Object matching with group filtering techniques

In this subsection, we present our object matching algorithm and our proposed group filtering techniques. The high-level of our object matching algorithm works as follow.

```
Algorithm 1: ObjectMatching
Data: New geo-textual object $o_n$, Object set $Q$, HGI index $\mathcal{I}(Q)$, similarity
      threshold $\theta$, Current join result set $\mathcal{S}$
Result: Update of join result set $\mathcal{S}$
1  for each cell $c$ in $\mathcal{I}(Q)$ do
2      $ub \leftarrow Sim^{s}_{ub}(o_n, c)$;
3      if $ub \geq \theta$ then
4          for each bucket $B$ indexed under $c$ do
5              $ub_t \leftarrow Sim^{st}_{ub}(o_n, B)$;
6              if $ub_t \geq \theta$ then
7                  for each term $w$ in $o_n.\psi$ do
8                      $I_w \leftarrow$ postings list of $w$;
9                      for each posting $p$ of $I_w$ do
10                         $o_i \leftarrow p.entry$;
11                         $s \leftarrow Sim(o_n, o_i)$;
12                         if $s \geq \theta$ then
13                             $\mathcal{S}$.add($\langle o_n, o_i \rangle$);
14 return $\mathcal{S}$;
```

Algorithm 1 presents the pseudo code of our object matching algorithm. The inputs are (1) new geo-textual object o_n, (2) object set Q, (3) HGI index $\mathcal{I}(Q)$, of which the data being indexed is Q, (4) similarity threshold θ, and (5) current join result set $\mathcal{S}$. The output is the updated join result set that takes the object pairs related to o_n into consideration.

When a new object o_n arrives, we first evaluate o_n against the objects in each cell c. Specifically, for each cell c, we calculate the similarity upper bound between o_n and objects indexed by c, which is denoted by $Sim^s_{ub}(o_n, c)$ (Line 2). The value of $Sim^s_{ub}(o_n, c)$ is computed by (5).

$$Sim^s_{ub}(o_n, c) = \alpha \times \left(1 - \frac{dist_{min}(o_n.\rho, c)}{dist_{max}}\right) + \beta + \gamma, \tag{5}$$

where $dist_{min}(o_n.\rho,c)$ denotes the minimum Euclidean distance between $o_n.\rho$ and c. If the similarity upper bound between o_n and objects indexed by c is no less than the similarity threshold θ (Line 3), we proceed to evaluate o_n against each bucket indexed under c. To be specific, for each bucket B indexed under cell c, we calculate the similarity upper bound between o_n and objects indexed by bucket B, which is denoted by $Sim^{st}_{ub}(o_n, B)$ (Line 5). The value of $Sim^{st}_{ub}(o_n, B)$ is computed by (6).

$$Sim^{st}_{ub}(o_n, B) = \alpha \times \left(1 - \frac{dist_{min}(o_n.\rho, c)}{dist_{max}}\right) + \gamma \times \left(1 - \frac{|o_n.t - B.t_{max}|}{t_{current} - t_{earliest}}\right) + \beta, \tag{6}$$

where $B.t_{max}$ denotes the latest timestamp of objects in B. If the similarity upper bound between o_n and objects indexed by bucket B is no less than the similarity threshold θ (Line 6), we proceed to visit the inverted file maintained by B. In particular, for each term w in $o_n.\psi$ we retrieve its corresponding postings list, which is denoted by I_w (Line 8). For each posting p in I_w we retrieve its entry and get the corresponding object o_i through the entry (Line 10). After that, we compute the exact similarity between o_n and o_i. If the similarity between o_n and o_i is no less than θ, we add object pair (o_n,o_i) into the join result set $\mathcal{S}$ (Lines 12–13). Finally, we return $\mathcal{S}$ as the updated join result (Line 14).

5 Experimental study

This section presents our detailed experimental studies regarding the performance of our proposal.

5.1 Baseline

Our baseline approach is presented in Section 4.1. It is basically a brute force search algorithm. Each time a new geo-textual object arrives, we compute the similarity between the new object and every existing object. The baseline approach is named as Brute Force Matching Algorithm, which is abbreviated as BFM.

5.2 Datasets

We use two real-life geo-textual datasets in our experiments, namely FQ and TE. We proceed to introduce the two datasets respectively. FQ is a real-life dataset collected from Foursquare. The dataset consists of 2 million check-ins from all over the world. Each check-in contains a spatial coordinate with latitude and longitude, and a short text description. As for TE, it is also a real-life dataset. It is collected from Twitter. It consists of 10 million tweets with geographical locations. Likewise, each tweet contains a spatial coordinate and a short text (up to 140 characters).

5.3 Experimental settings

Our parameter settings are denoted by Table 1. It is worthy of noting that we use BFM to represent the baseline approach, which is the abbreviation of brute force matching algorithm. We use TriFilter to represent our proposed method with both HGI indexing structure and group filtering techniques applied.

5.4 Experimental result

Our experimental results are presented as follow.

5.4.1 Effect of the number of object keywords

The first set of experiments evaluates the effect of the number of keywords in each geo-textual object. Figure 3 shows the performance results of BFM and TriFilter on FQ and TE datasets respectively. We see that both methods exhibit an increasing trend regarding the processing time when we increase the number of object keywords. The reason can be explain by the fact that when the number of object keywords increases, we need to visit more terms and postings. As such, it may take more time to retrieve the terms and postings.

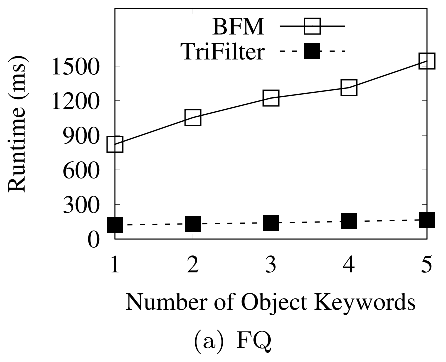
In addition, we find that the runtime increasing trend of BFM is more significant than the runtime increasing trend of TriFilter. The reason is that TriFilter may help filter out some unqualified pairs at an early stage.

5.4.2 Effect of the number of grid cells

Figure 4 demonstrates the results on FQ and TE respectively when we vary the number of grid cells regarding TriFilter. We see that when we increase the number of grid cells from 16 to 256, the object matching runtime performance becomes better. The reason is that more grid cells in the HGI index may indicate that the objects are more likely to be filtered out in a group manner. As such, more unqualified objects may be pruned at an early stage without the need of evaluating individually. However, when we proceed to increase the number of grid cells from 256 to 1024, we see that the runtime

Table 1 Parameter settings

Parameter	Setting	Default
Number of object keywords	1 – 5	3
Number of grid cells	16 – 1024	256
Parameter α	0.2 – 0.8	0.33
Parameter β	0.2 – 0.8	0.33
Similarity threshold θ	0.90 – 0.98	0.94
Number of indexed objects	based on situations	FQ:1M TE 5M



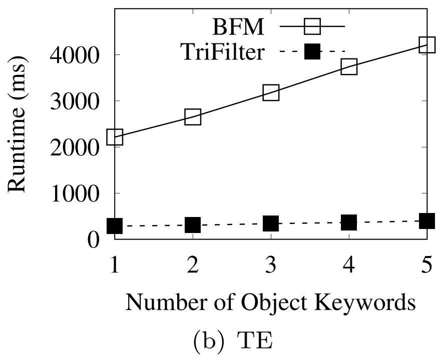
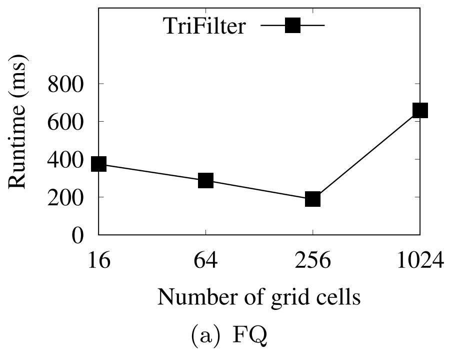


Figure 3 Effect of the number of object keywords



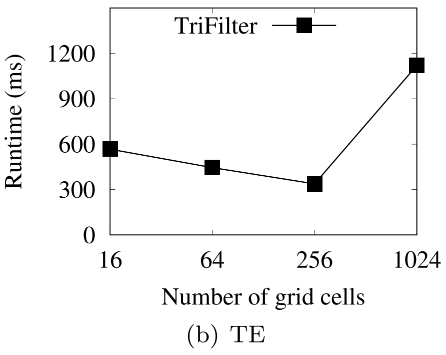


Figure 4 Effect of the number of grid cells

increases significantly. The reason is that more grid cells, in turn, may increase the number of computations regarding the similarity upper bound between a new object and a group of objects in a cell. Moreover, we find that the TriFilter performs best when the number of grid cells is set to be 256 on both FQ and TE.

5.4.3 Effect of spatial weight parameter

Figure 5 illustrates the performance comparison between BFM and TriFilter when we tune the spatial weight parameter α from 0.2 to 0.8. Note that when we tune the weight parameter α, the other two weight parameters β and γ are set to be equal. For example, when α is set to be 0.2, β and γ are both set to be $\frac{1-0.2}{2} = 0.4$. For TriFilter, we see that the time cost of object matching decreases when we increase the value of α. The reason is that when we increase the value of α, spatial proximity between two objects will be weighted more. As such, indexed objects are more likely to be pruned in the first phase. As for BFM, the runtime exhibits a relatively consistent trend as we vary the value of α.

5.4.4 Effect of textual weight parameter

Figure 5 demonstrates the performance comparison between BFM and TriFilter as we vary the textual weight parameter β from 0.2 to 0.8. Note that when we tune the weight parameter β, the other two weight parameters α and γ are set to be equal. For instance, when β is set to be 0.6, α and γ are both set to be $\frac{1-0.6}{2} = 0.2$. For TriFilter, we see that the time cost of object matching increases when we increase the value of β. The reason is that when we increase the value of β, spatial proximity between two objects will be weighted less. As such, indexed objects are less likely to be pruned in the first phase, which may induce more checking and evaluation. As for BFM, the runtime exhibits a relatively consistent trend as we vary the value of β (Figure 6).

5.4.5 Effect of similarity threshold

In the last set of experiments, we evaluate the object matching performance when we vary the similarity threshold θ. From Figure 7 we see that the runtimes of both methods exhibit an decreasing trend as we increase the similarity threshold θ. The reason can be explained by the fact that when the similarity threshold becomes greater, more indexed objects may be pruned because more object pairs may become unqualified. Further, we notice that the runtime decreasing trend regarding TriFilter is more conspicuous than that of BGM. The reason is that TriFilter has more group filtering mechanisms. Higher value of similarity threshold may make a group of objects be more likely to be filtered out.

6 Related work

In this section, we investigate related studies regarding geo-textual object matching and location-based similarity join.

6.1 Matching of geo-textual objects

The problem of geo-textual object matching can be classified into two categories: geo-textual data search and similarity join of geo-textual data. The problem of geo-textual

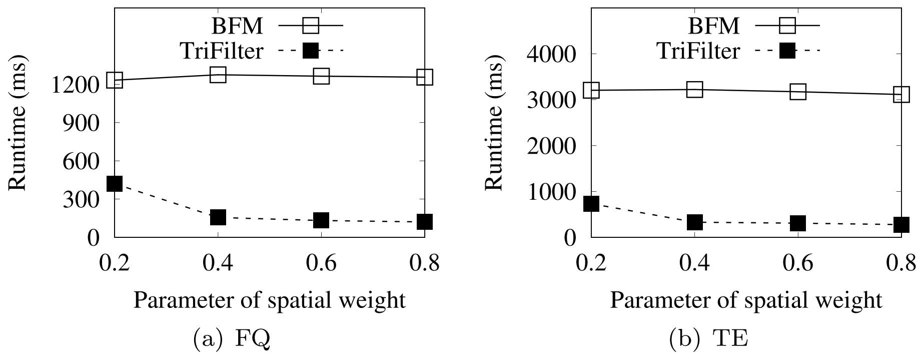
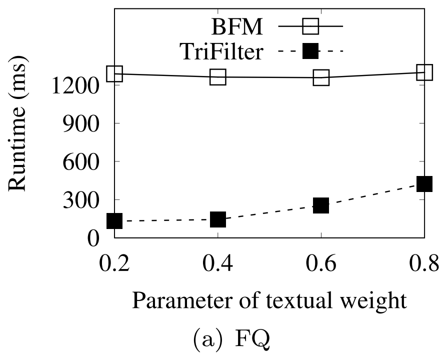


Figure 5 Effect of spatial weight α



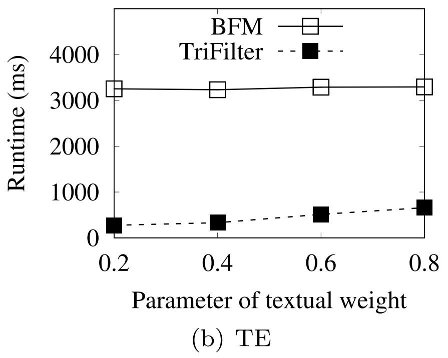
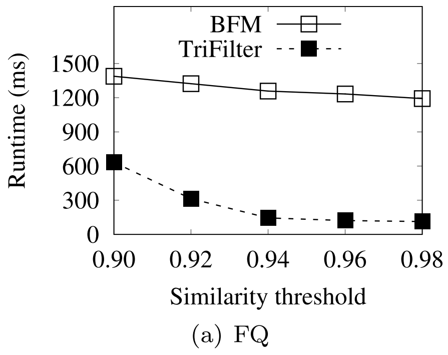


Figure 6 Effect of textual weight β

search requires users to define spatial keyword queries. Each spatial keyword query may have spatial requirement and textual requirement. Spatial requirement may be defined as a distance threshold, s spatial proximity score, or a spatial region. Textual requirement may be defined by a set of query keywords or a keyword-based Boolean predicate. The problem of spatial keyword query processing has been extensively studied by existing literature [1–9]. In particular, some studies take the semantic meaning of geo-textual objects into consideration [10, 11]. Sequential geo-textual data query processing is also investigated by existing literature regarding trajectory data analytics [12]. A number of benchmark studies and surveys have been published over the past few years [13–15]

Recently, the problem of continuous spatial keyword search has been investigated as well. They define the problem of continuous spatial keyword search in a variety of fashions, including location-based pubish/subscribe [16–24], continuous geo-textual filtering mechanism [25], and moving geo-textual object processing [26].

Nonetheless, the aforementioned studies define the problem as query processing problems, which cannot be directly used to handle the problem of geo-textual similarity join.



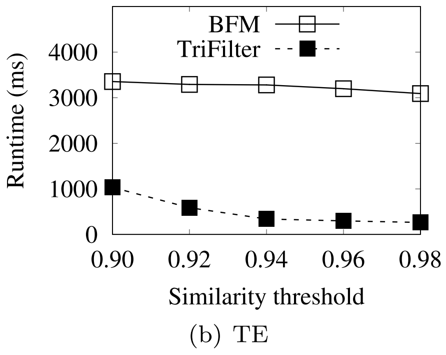


Figure 7 Effect of similarity threshold θ

6.2 Location based similarity join

Existing studies regarding the problem of location-based similarity join can be classified into the following categories. The first category is geo-textual object similarity join [27, 28]. It is regarding the join operation between two sets of geo-textual objects. The second category is trajectory similarity join [29–33]. It is regarding the join operation between two sets of trajectory data. The third category is trajectory to object similarity join [34]. It is regarding the join operation between a set of trajectories and a set of geo-located objects.

However, existing similarity join studies has the following limitations. First, they do not consider the popular scenario where the dataset is updated in a continuous manner. Second, they do not consider all of the three major aspects for geo-textual objects: spatial proximity, text relevancy, and temporal gap. As a result, they are inapplicable to our CGTS-Join problem.

7 Conclusions

We consider the problem of Continuous Geo-Textual Similarity Join (CGTS-Join). To this end, we define a new geo-textual similarity metric to measure the similarity between two geo-textual objects by taking spatial proximity, textual relevancy, and temporal gap into consideration. We develop a Hybrid Grid Indexing Structure (HGI) to effectively organize massive-scale geo-textual objects. We also propose a Tri-Filtering framework to answer the CGTS-Join problem. The experimental results on two real-life datasets show that our proposal, Tri-Filtering framework, is capable of achieving a runtime reduction of 60%-90% in comparison against 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 contributions 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

Ethics approval and consent to participate Not applicable

Consent for publication Not applicable

Human and animal ethics Not applicable

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

References

1. Felipe, I.D., Hristidis, V., Rishe, N.: Keyword search on spatial databases. In: ICDE, pp 656–665 (2008)
2. Cong, G., Jensen, C.S., Wu, D.: Efficient retrieval of the top-k most relevant spatial web objects. In: PVLDB, pp 337–348 (2009)
3. Rocha-Junior, J.B., Orestis, G., Simon, J., Nørvåg, K.: Efficient processing of top-k spatial keyword queries. In: SSTD, pp 205–222 (2011)
4. Zhang, D., Tan, K.-L., Tung, A.K.H.: Scalable top-k spatial keyword search. In: EDBT, pp 359–370 (2013)
5. Zhang, C., Zhang, Y., Zhang, W., Lin, X.: Inverted linear quadtree: Efficient top k spatial keyword search. In: ICDE, pp 901–912 (2013)
6. 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)
7. Yang, C., Chen, L., Shang, S., Zhu, F., Li, L., Shao, L.: Toward efficient navigation of massive-scale geo-textual streams. In: IJCAI, pp 4838–4845 (2019)
8. 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)
9. Xu, J., Sun, J., Zhou, R., Liu, C., Yin, L.: CISK: An interactive framework for conceptual inference based spatial keyword query. Neurocomputing **428**, 368–375 (2021)
10. Chen, X., Xu, J., Zhou, R., Zhao, P., Liu, C., Fang, J., Zhao, L.: S^{2}r-tree: A pivot-based indexing structure for semantic-aware spatial keyword search. GeoInformatica **24**(1), 3–25 (2020)
11. Qian, Z., Xu, J., Zheng, K., Zhao, P., Zhou, X.: Semantic-aware top-k spatial keyword queries. World Wide Web **21**(3), 573–594 (2018)
12. Song, X., Xu, J., Zhou, R., Liu, C., Zheng, K., Zhao, P., Falkner, N.J.G.: Collective spatial keyword search on activity trajectories. GeoInformatica **24**(1), 61–84 (2020)
13. Chen, L., Cong, G., Jensen, C.S., Wu, D.: Spatial keyword query processing: an experimental evaluation. In: PVLDB, pp 217–228 (2013)
14. Chen, L., Shang, S., Yang, C., Li, J.: Spatial keyword search: a survey. GeoInformatica **24**(1), 85–106 (2020)
15. 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)
16. Li, G., Wang, Y., Wang, T., Feng, J.: Location-aware publish/subscribe. In: KDD, pp 802–810 (2013)
17. Chen, L., Shang, S., Jensen, C.S., Xu, J., Kalnis, P., Yao, B., Shao, L.: Top-k term publish/subscribe for geo-textual data streams. VLDB J. **29**(5), 1101–1128 (2020)
18. Chen, L., Shang, S.: Approximate spatio-temporal top-k publish/subscribe. World Wide Web **22**(5), 2153–2175 (2019)
19. 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)
20. 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)
21. 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)
22. 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)
23. Hu, H., Liu, Y., Li, G., Feng, J., Tan, K.-L.: A location-aware publish/subscribe framework for parameterized spatio-textual subscriptions. In: ICDE, pp 711–722 (2015)
24. Chen, L., Cong, G., Cao, X., Tan, K.-L.: Temporal spatial-keyword top-k publish/subscribe. In: ICDE, pp 255–266 (2015)
25. Chen, L., Cong, G., Cao, X.: An efficient query indexing mechanism for filtering geo-textual data. In: SIGMOD, pp 749–760 (2013)
26. Guo, L., Zhang, D., Li, G., Tan, K.-L., Bao, Z.: Location-aware pub/sub system: When continuous moving queries meet dynamic event streams. In: SIGMOD, pp 843–857 (2015)
27. Amagata, D., Tsuruoka, S., Arai, Y., Hara, T.: Feat-sksj: Fast and exact algorithm for top-k spatial-keyword similarity join. In: SIGSPATIAL '21: 29th International Conference on Advances in Geographic Information Systems, Virtual Event / Beijing, China, November 2-5, 2021, pp 15–24. ACM (2021)
28. Hu, H., Li, G., Bao, Z., Feng, J., Wu, Y., Gong, Z., Xu, Y.: Top-k spatio-textual similarity join. IEEE Trans. Knowl. Data Eng. **28** (2), 551–565 (2016)
29. Ta, N., Li, G., Xie, Y., Li, C., Hao, S., Feng, J.: Signature-based trajectory similarity join. IEEE Trans. Knowl. Data Eng. **29**(4), 870–883 (2017)

30. Yuan, H., Li, G.: Distributed in-memory trajectory similarity search and join on road network. In: 35th IEEE International Conference on Data Engineering, ICDE 2019, Macao, China, April 8-11, 2019, pp 1262–1273. IEEE (2019)
31. Shang, S., Chen, L., Wei, Z., Jensen, C.S., Zheng, K., Kalnis, P.: Trajectory similarity join in spatial networks. Proc VLDB Endow. **10**(11), 1178–1189 (2017)
32. Shang, S., Chen, L., Wei, Z., Jensen, C.S., Zheng, K., Kalnis, P.: Parallel trajectory similarity joins in spatial networks. VLDB J. **27**(3), 395–420 (2018)
33. Chen, L., Shang, S., Jensen, C.S., Yao, B., Kalnis, P.: Parallel semantic trajectory similarity join. In: 36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, April 20-24, 2020, pp 997–1008. IEEE (2020)
34. Shang, S., Chen, L., Zheng, K., Jensen, C.S., Wei, Z., Kalnis, P.: Parallel trajectory-to-location join. IEEE Trans. Knowl. Data Eng. **31**(6), 1194–1207 (2019)