A Prefix-Filter based Method for Spatio-Textual Similarity Join

Sitong Liu, Guoliang Li, and Jianhua Feng

Abstract—Location-based services have attracted significant attention due to modern mobile phones equipped with GPS devices. These services generate large amounts of spatio-textual data which contain both spatial location and textual descriptions. Since a spatio-textual object may have different representations, possibly because of deviations of GPS or different user descriptions, it calls for efficient methods to integrate spatio-textual data from different sources. In this paper we study a new research problem called spatio-textual similarity join: given two sets of spatio-textual objects, find the similar object pairs. We make the following contributions: (1) We develop a filter-and-refine framework and devise several efficient algorithms. We extend the prefix filter technique to generate spatial and textual signatures for the objects and build inverted index on top of these signatures. Then we generate candidate pairs using the inverted lists of signatures. Finally we refine the candidates and generate the final result. (2) We study how to generate high-quality signatures for spatial information. We develop an MBR-prefix based signature to prune large numbers of dissimilar object pairs. (3) We propose a hybrid signature scheme to support both textual pruning and spatial pruning simultaneously. (4) Experimental results on real and synthetic datasets show that our algorithms achieve high performance and scale well.

Index Terms—Spatio-Textual Objects, Similarity Join, MBR Prefix, Hybrid Signature

1 INTRODUCTION

With the near ubiquity of global position systems (GPS) in smartphones, location-based services (LBS) have recently attracted significant attentions from both academic and industrial community. These services generate large amounts of spatio-textual data which contain both geographical location and textual description. As a spatio-textual object may have different representations, possibly because of deviations of GPS or different user descriptions, it calls for efficient methods to correlate the spatio-textual data from different sources. For example, Google Map[1] generates the detailed information of points of interests (e.g., hotels and restaurants) by integrating the relevant data from multiple sources. Factual[2] extracts spatio-textual information from the user-generated data to generate new points of interest.

In this paper, we study a new research problem, called Spatio-textual similarity join (`StarJoin`). Given two sets of spatio-textual objects with a spatial region and textual descriptions, it finds all similar object pairs. Two objects are similar if their spatial similarity and textual similarity are larger than given thresholds. In this paper, we use Jaccard coefficient as an example to quantify the spatial similarity and textual similarity and our techniques can be easily extended to support other similarity functions. `StarJoin` has many real applications. One example is Tuan800[3], a famous information integration service which integrates discount information from various group-on websites. Each discount message is associated with a spatial range and some keyword descriptions. Since different group-on websites may contain many similar discount messages, it is very important to perform a similarity join on the datasets so as to eliminate redundant ones for improving user experiences. Another example is message delivery. For instance, Twitter[4] allows users to upload their locations while posing tweets and thus we can define users' active regions using their locations (e.g., cities). We can use existing keyword extraction techniques to capture users' interests (e.g., travel, jogging). We can model users as spatio-textual objects. Meanwhile, twitter messages (e.g., coupons, ads) can also be modelled as spatio-textual objects. To deliver messages to relevant users, we can perform a similarity join on the user set and the message set.

There are some recent studies on spatial join [18], [15], [20], [4] and string similarity join (for textual data)[6], [1]. Although we can extend their methods to support our problem (see Section 2.2), they are rather inefficient as they only use spatial pruning or textual pruning, and may generate large numbers of intermediate results. To address this limitation and improve the performance, we develop a signature based filter-and-refine framework. First we generate spatial and textual signatures for the objects and build inverted indexes to avoid redundant computations. Then we use these signatures to find candidate pairs whose signatures are similar enough. Finally we verify the candidates

• *Sitong Liu, Guoliang Li and Jianhua Feng are with the Department of Computer Science and Technology, Tsinghua National Laboratory for Information Science and Technology (TNList), Tsinghua University, Beijing 100084, China. E-mail: liu-st10@mails.tsinghua.edu.cn; {liguoliang, fengjh}@tsinghua.edu.cn*

1. http://maps.google.com
2. http://www.factual.com
3. http://www.tuan800.com
4. http://twitter.com

to get the final answers. We propose several algorithms by organizing spatial and textual signatures in different ways. In addition, we propose two effective methods to generate high-quality signatures. The first one selects a subregion as a signature for each object. We prove that the subregion selected in our method is minimized. The second method is a hybrid signature which integrates spatial information and textual descriptions. We also develop effective pruning techniques to improve the performance.

We summarize our main contributions as follows.

(1) We study a new research problem called Spatio-textual Similarity Join. We explore a filter-and-refine framework and propose efficient algorithms which can prune large numbers of dissimilar objects.

(2) We develop an MBR-Prefix based signature which uses subregions of objects as signatures to support spatial pruning. We prove that the selected subregion is minimized.

(3) We propose a hybrid signature by integrating spatial and textual pruning. To optimize the hybrid signature, we model the problem as a token partition problem and prove it is NP-complete. We develop a greedy partition algorithm and integrate it into our algorithms.

(4) We have conducted extensive experiments on real and synthetic datasets. Experimental results show that our methods achieve high performance and scale well.

The rest of this paper is organized as follows. We formulate our problem in Section 2 and propose a filter-and-refine framework in Section 3. Section 4 and Section 5 devise two effective filtering techniques respectively. Experimental results are provided in Section 6. We review related work in Section 7 and make a conclusion in Section 8.

2 Preliminaries

We first formulate the problem of spatio-textual similarity join in Section 2.1, and then show possible solutions using state-of-the-art algorithms in Section 2.2.

2.1 Problem Statement

Consider two collections of objects $\mathcal{R} = \{r_1, r_2 ..., r_n\}$ and $\mathcal{S} = \{s_1, s_2 ..., s_m\}$. Each object r (or s) includes a spatial region $\mathcal{M}_r$ and textual description $\mathcal{T}_r$. In this paper we use Minimum Bounding Rectangle (MBR) to capture the spatial information, denoted by $\mathcal{M}_r = [r_{bl}, r_{tr}]$, where $r_{bl} = (r_{bl}.x, r_{bl}.y)$ is the bottom-left point and $r_{tr} = (r_{tr}.x, r_{tr}.y)$ is the top-right point. We use a set of tokens to capture the textual description, denoted by $\mathcal{T}_r = \{t_1, t_2, \ldots, t_v\}$, which describes an object (e.g., {`Hotel, Pizza, Subway`}) or users' interests (e.g., {`Seaside, Sandwich, Delivery`}). As tokens may have different importance, we assign each token t_i with a weight $w(t_i)$ (e.g., inverse document frequency `idf`).

To quantify the similarity between two objects, we use the well-known Jaccard as an example to evaluate the spatial similarity (S_{Jac}) and textual similarity (T_{Jac}). Our techniques can be easily extended to support other similarity functions. We will discuss the details in Section 5.2.

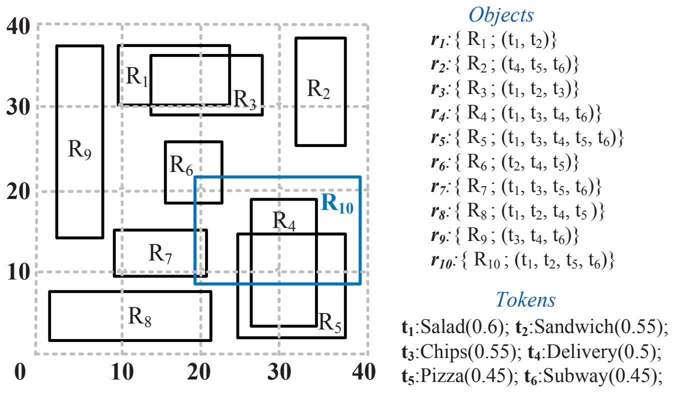

40
30
20
10
0
10
20
30
40
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
Objects
r1: { R1 ; (t1, t2)}
r2: { R2 ; (t4, t5, t6)}
r3: { R3 ; (t1, t2, t3)}
r4: { R4 ; (t1, t3, t4, t6)}
r5: { R5 ; (t1, t3, t4, t5, t6)}
r6: { R6 ; (t2, t4, t5)}
r7: { R7 ; (t1, t3, t5, t6)}
r8: { R8 ; (t1, t2, t4, t5)}
r9: { R9 ; (t3, t4, t6)}
r10: { R10 ; (t1, t2, t5, t6)}
Tokens
t1:Salad(0.6); t2:Sandwich(0.55);
t3:Chips(0.55); t4:Delivery(0.5);
t5:Pizza(0.45); t6:Subway(0.45);

Fig. 1. An example of spatio-textual objects

Definition 1 (Spatial Jaccard): Given two objects r and s, their spatial Jaccard similarity (S_{Jac}) is defined as:

$$S_{Jac}(r, s) = \frac{|\mathcal{M}_r \cap \mathcal{M}_s|}{|\mathcal{M}_r| + |\mathcal{M}_s| - |\mathcal{M}_r \cap \mathcal{M}_s|}$$

where $|\cdot|$ is the size of an MBR.

Definition 2 (Textual Jaccard): Given two objects r and s, their textual Jaccard similarity (T_{Jac}) is defined as:

$$T_{Jac}(r, s) = \frac{\sum_{t \in \mathcal{T}_r \cap \mathcal{T}_s} w(t)}{\sum_{t \in \mathcal{T}_r \cup \mathcal{T}_s} w(t)}$$

where $w(t)$ is the weight of token t.

Two objects r and s are similar if they satisfy (1) Spatial constraint: their spatial Jaccard similarity is larger than a spatial similarity threshold τ_s, i.e., $S_{Jac}(r, s) > \tau_s$; and (2) Textual constraint: their textual Jaccard similarity is larger than a textual similarity threshold τ_t, i.e., $T_{Jac}(r, s) > \tau_t$. We formulate the spatio-textual similarity join problem.

Definition 3 (Spatio-Textual Similarity Join): Given two collections of objects $\mathcal{R} = \{r_1, r_2 ..., r_n\}$, $\mathcal{S} = \{s_1, s_2 ..., s_m\}$, and two similarity thresholds τ_s and τ_t, a spatial-textual similarity join finds all similar pairs (r_i, s_j) where $S_{Jac}(r_i, s_j) > \tau_s$ and $T_{Jac}(r_i, s_j) > \tau_t$.

Example 1: Consider the ten objects in Figure 1. Suppose $\tau_s = 0.6$ *and* $\tau_t = 0.7$. *For object pair* (r_1, r_3) *where* $R_1 = [(10, 30), (24, 37)]$ *and* $R_3 = [(11, 29), (26, 36)]$, $S_{Jac}(r_1, r_3) = \frac{13\times6}{14\times7+15\times7-13\times6} = 0.624$ *and* $T_{Jac}(r_1, r_3) = \frac{0.6+0.55}{0.6+0.55+0.55} = 0.676$. *Thus,* r_1 *and* r_3 *are not similar since* $0.676 < \tau_t$. *Consider another pair*(r_4, r_5) *where* $R_4 = [(27, 4), (35, 19)]$ *and* $R_5 = [(25, 4), (37, 17)]$. S_{Jac} $= 0.61$ *and* $T_{Jac} = 0.82$. *Since* $0.61 > \tau_s$ *and* $0.82 > \tau_t$, (r_4, r_5) *is an answer.*

Discussions: For ease of presentation, we make the following hypothesis: (1) $\mathcal{R} = \mathcal{S}$. We will extend our techniques to support the case $\mathcal{R} \neq \mathcal{S}$ in Section 5.2; (2) Each object has a distinct MBR and an MBR can be taken as an identifier of an object. We use the object ID r_i and its MBR R_i interchangeably to denote the object. Our methods can also support multiple objects with the same MBR.

2.2 Straightforward Methods

In this section we first introduce plane sweep algorithm and prefix filter technique. Then we show how to extend these methods to support our problem.

Plane Sweep: The plane sweep algorithm [21] is a well-known method for spatial join problems. The basic idea is that if two MBRs have overlap, their projections along $x(y)$ axis must also have overlap. Suppose all the objects are sorted along one axis, e.g., x. We use a vertical line l sweeping through the entire dataset from right to left. At any point, the objects intersecting with l have common intervals along x-axis. We regard these pairs as candidates and refine them using the other axis (e.g., y).

Prefix Filter: Prefix filter [6] is a widely used method for keyword similarity join problem. Its basic idea is that if the similarity of two token sets is larger than a given threshold, they should share enough common tokens. Then we can reduce the index by cutting off some common tokens. For simplicity, we first suppose $w(t_i) = 1$. Consider two token sets $\mathcal{T}_r$ and $\mathcal{T}_s$. Since $|\mathcal{T}_r \cap \mathcal{T}_s|$ is an integer, according to [6], if $T_{Jac}(\mathcal{T}_r, \mathcal{T}_s) > \tau_t$, then $|\mathcal{T}_r \cap \mathcal{T}_s| > \tau_t \times |\mathcal{T}_r| \geq \lfloor \tau_t \times |\mathcal{T}_r| \rfloor + 1$. Based on this property, we first sort all the tokens according to a global ordering, e.g., `idf`. Then for each token set $\mathcal{T}$, we generate its prefix by deleting last $\lfloor \tau_t \cdot |\mathcal{T}| \rfloor$ tokens. If any two objects have common tokens in their prefixes, we add them into the candidate set. Next we extend the prefix filter technique to the case $w(t_i) \neq 1$ (the weight may not be an integer). We first sort tokens in the descending order of their weights. Since $|\mathcal{T}_r \cap \mathcal{T}_s| > \tau_t \times \sum_{i=1}^{|\mathcal{T}_r|} w(t_i)$, then for each token set $\mathcal{T}$, we generate its prefix by deleting the last k tokens which satisfy $\sum_{i=|\mathcal{T}|-k+1}^{|\mathcal{T}|} w(t_i) \leq \tau_t \cdot \sum_{i=1}^{|\mathcal{T}|} w(t_i)$ and $\sum_{i=|\mathcal{T}|-k}^{|\mathcal{T}|} w(t_i) > \tau_t \cdot \sum_{i=1}^{|\mathcal{T}|} w(t_i)$.

Example 2: Suppose $\tau_t = 0.7$. *Consider* $r_1 = \{t_1, t_2\}$ *and* $r_3 = \{t_1, t_2, t_3\}$. *The weights of* t_1, t_2, t_3 *are respectively* $0.6, 0.55, 0.55$. *For* r_1, *since* $0.7 \times (0.6 + 0.55) = 0.805$, *we can delete at most last k tokens whose weight sum is no larger than* 0.805. *Thus,* t_2 *is deleted and* $Prefix(r_1) = \{t_1\}$. *Similarly,* $Prefix(r_3) = \{t_1\}$. *Since* $Prefix(r_1) \cap Prefix(r_3) \neq \phi$, (r_1, r_3) *is a candidate. Similarly,* $Prefix(r_8) = \{t_1, t_2\}$, $Prefix(r_9) = \{t_3\}$. (r_8, r_9) *is not a candidate since their prefixes do not share common tokens.*

Based on these two state-of-the-art techniques, two kinds of possible solutions are under consideration: (1) **Textual-first method:** It first uses existing textual similarity join methods, e.g., Prefix filter [2], to prune the pairs that do not satisfy the textual constraint. Then it refines the candidate pairs with spatial constraints. In this paper, we use PPJoin [26], a state-of-the-art prefix-filtering algorithm, as an example and all other existing methods can be applied to our algorithms. (2) **Spatial-first method:** It first uses prevalent spatial join algorithms, e.g., Plane sweep [21], to generate candidate pairs which satisfy the spatial constraint. Then the candidates are refined using textual constraints. In this paper, we use the plane sweep algorithm. Obviously, these methods will generate significant numbers of candidates since only one constraint is used in the filter step.

```
Algorithm 1: Star-Join Framework ($\mathcal{R}, \tau_s, \tau_t$)
  Input: $\mathcal{R}$: An objects set; $\tau_s$: Spatial threshold
         $\tau_t$: Textual threshold
  Output: $\mathcal{P}$: a set of similar pairs
1 begin
2     $\mathcal{I} \leftarrow$ empty index;
3     $\mathcal{C} \leftarrow$ empty candidate set;
4     for each object $r \in \mathcal{R}$ do
5         SIG($r$) ←SigGeneration ($r$, $\tau_s$, $\tau_t$);
6         $\mathcal{C}$ ←CandidateFiltering (SIG($r$), $\mathcal{I}$);
7         $\mathcal{I}$ ←IndexUpdating (SIG($r$), $\mathcal{I}$);
8         $\mathcal{P}$ ←Refinement ($\mathcal{C}$);
9 end
```

Fig. 2. A Filter-and-Refine framework

3 PREFIX FILTER BASED METHODS

In this section, we first propose a filter-and-refine framework in Section 3.1 and devise several filtering algorithms in Section 3.2. The discussion of sorting strategies and complexity analysis will be given in Section 3.3 and Section 3.4.

3.1 A Filter-and-Refine Framework

To avoid enumerating every object pair, we introduce an incremental signature-based framework. We scan the objects in order (the sorting strategies will be discussed in Section 3.3) and build index for all the objects that have been visited. For current object r, we first find all the candidates of r using the index and then insert the signature of r into the index. Iteratively, we can find all candidate pairs. Finally we verify the candidates to generate the final result. Our framework includes three steps shown in Algorithm 1.

Filter: We generate the signatures of objects and use these signatures to get the candidates. In this paper, the signatures should satisfy the following property. Given two objects r and s, let SIG(r) and SIG(s) respectively denote their signatures. If r and s are similar, then SIG(r) $\cap$ SIG(s) $\neq \phi$. We sequentially scan the objects and maintain an inverted index for signatures of the visited objects. For the current object r, we generate its signature (Line 5) and use the inverted index to find its candidates (Line 6).

Index Update: After finding all the candidates of current object r, we insert the signature of r to the end of current index (Line 7).

Refine: We refine all the candidate pairs and check whether they satisfy spatial and textual constraints simultaneously by computing their real similarity (Line 8). In this paper, we mainly focus on the filter and the index update step.

3.2 Generating Spatial Prefixes

In our framework, there are two main challenges: (1) how to generate high quality signatures using spatial and textual information; (2) how to organize these signatures when building the index.

Signature Generation: Given an object r, we use its token prefix $\mathcal{T}_r$ as its textual signature (Section 2.2), denoted by $\textsc{Sig}_T(r)$. Now we discuss how to generate spatial signatures. To estimate the region of an object r, a natural idea is to partition the space into grids and associate r with the grids that intersect with $\mathcal{M}_r$, denoted by $\mathcal{G}_r$. For each grid $g_i \in \mathcal{G}_r$, we denote its intersection with $\mathcal{M}_r$ as $r(g_i) = |\mathcal{M}_{g_i} \cap \mathcal{M}_r|$ where $\mathcal{M}_{g_i}$ is the region of grid g_i. Recall the textual prefix filtering property in Section 2.2, we now extend it to support spatial data. Suppose objects r and s are similar. As $|\mathcal{M}_r \cap \mathcal{M}_s| \le \min(|\mathcal{M}_r|, |\mathcal{M}_s|)$, based on Definition 1, we can easily get *if r and s satisfy S_{Jac} constraint, their intersection area must be larger than* $\tau_s \times \max\{|\mathcal{M}_r|, |\mathcal{M}_s|\}$. We can utilize the prefix-filtering technique to generate spatial signatures as follows. Given an object r, we first sort grids in $\mathcal{G}_r$ based on a global order (The ordering strategies will be discussed in Section 3.3). Then we can generate its signature by deleting last k grids which satisfy $\sum_{i=|\mathcal{G}_r|-k+1}^{|\mathcal{G}_r|} |\mathcal{M}_{g_i} \cap \mathcal{M}_r| \le \tau_s \cdot |\mathcal{M}_r|$ and $\sum_{i=|\mathcal{G}_r|-k}^{|\mathcal{G}_r|} |\mathcal{M}_{g_i} \cap \mathcal{M}_r| > \tau_s \cdot |\mathcal{M}_r|$, denoted by $\textsc{Sig}_S(r)$. If r and s are similar, then they at least share one common grid in their spatial signatures as stated in Lemma 1.

Lemma 1: Given two objects r and s, $\textsc{Sig}_S(r)$ and $\textsc{Sig}_S(s)$ are spatial signatures of r and s. If $S_{Jac}(r,s) > \tau_s$, then $\textsc{Sig}_S(r) \cap \textsc{Sig}_S(s) \neq \phi$.

Proof: We prove it by contradiction. Suppose $S_{Jac}(r,s) > \tau_s$ and $\textsc{Sig}_S(r) \cap \textsc{Sig}_S(s) = \phi$. $\mathcal{G}_r$ and $\mathcal{G}_s$ are the grids intersecting with $\mathcal{M}_r$ and $\mathcal{M}_s$. Let $\textsc{Sig}_S(r) = \{g_1, g_2, ..., g_r\}$ and $\textsc{Sig}_S(s) = \{g_1, g_2, ..., g_s\}$. Since $\textsc{Sig}_S(r) \cap \textsc{Sig}_S(s) = \phi$, we have $g_r \neq g_s$. Suppose $g_r < g_s$, then $\textsc{Sig}_S(r) \cap \mathcal{G}_s = \phi$. Thus, only grids in $\mathcal{G}_r \setminus \textsc{Sig}_S(r)$ can intersect with $\mathcal{G}_s$. Then the maximum intersection area is $\sum_{i=r+1}^{|\mathcal{G}_r|} |\mathcal{M}_{g_i} \cap \mathcal{M}_r| \le \tau_s \cdot |\mathcal{M}_r|$ which contradicts the hypothesis. When $g_r > g_s$, the proof is similar. □

Example 3: We give an example of spatial signature generation. For ease of presentation, we suppose all the grids are sorted based on their IDs (other sorting orders will be discussed later). Take the object r_{10} in Figure 3 as an example. Its spatial region R_{10} intersects with nine grids $\{g_2(2), g_3(20), g_4(18), g_6(10), g_7(100), g_8(90), g_{10}(3), g_{11}(30), g_{12}(27)\}$, *where the number in each bracket is the overlap size with $\mathcal{M}_{r_{10}}$. Suppose $\tau_s = 0.6$. For grid g, we use $r_{10}(g)$ to denote its intersection with $\mathcal{M}_{r_{10}}$. Based on our analysis, we can delete grids, the sum of whose overlap size is no more than $\tau_s \times \sum_{g \in \mathcal{G}_{r_{10}}} r_{10}(g) = 168$. Since $r_{10}(g_8) + r_{10}(g_{10}) + r_{10}(g_{11}) + r_{10}(g_{12}) = 150 \le 168$ and $r_{10}(g_7) + r_{10}(g_8) + r_{10}(g_{10}) + r(g_{11}) + r_{10}(g_{12}) = 250 > 168$, we can prune $g_8, g_{10}, g_{11}, g_{12}$ and $\textsc{Sig}_S(r_{10}) = \{g_2, g_3, g_4, g_6, g_7\}$.*

Signature Organization: We study how to organize these signatures and discuss six possible solutions.

(1) Spatial Only: We only use the spatial signatures to build inverted index. Each entry in the index is a grid which maintains a list of objects overlapping with the grid. For the current object r, we use its spatial signature $\textsc{Sig}_S(r)$ to probe the corresponding inverted lists. All the objects in the lists will be taken as candidates.

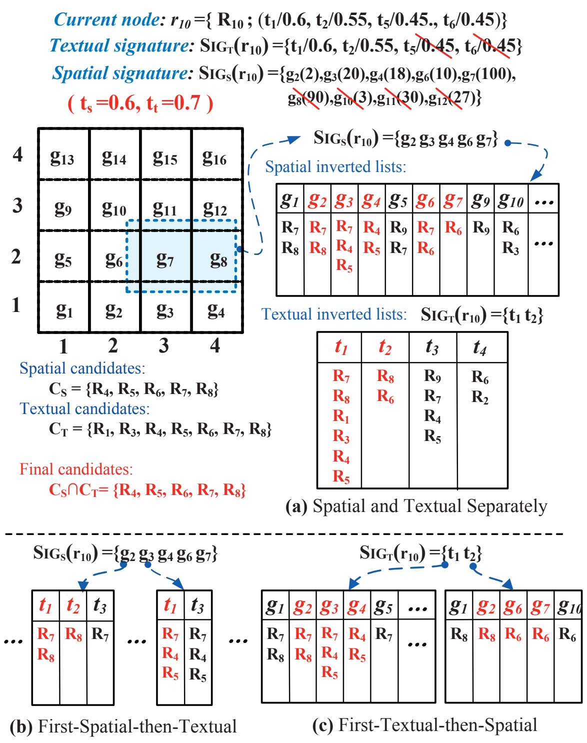

Current node: r10 ={ R10 ; (t1/0.6, t2/0.55, t5/0.45., t6/0.45)}
Textual signature: SigT(r10) ={t1/0.6, t2/0.55, ~~t5/0.45~~, ~~t6/0.45~~}
Spatial signature: SigS(r10) ={g2(2),g3(20),g4(18),g6(10),g7(100), ~~g8(90)~~,~~g10(3)~~,~~g11(30)~~,~~g12(27)~~}
(ts=0.6, tt=0.7)
g13 g14 g15 g16
g9 g10 g11 g12
g5 g6 g7 g8
g1 g2 g3 g4
SigS(r10) ={g2 g3 g4 g6 g7}
Spatial inverted lists:
Textual inverted lists: SigT(r10) ={t1 t2}
Spatial candidates: CS = {R4, R5, R6, R7, R8}
Textual candidates: CT = {R1, R3, R4, R5, R6, R7, R8}
Final candidates: CS∩CT= {R4, R5, R6, R7, R8}
(a) Spatial and Textual Separately
(b) First-Spatial-then-Textual
(c) First-Textual-then-Spatial

Fig. 3. Three prefix-filter based solutions

(2) Textual Only: We only use the textual signatures to build inverted index. Each entry in the index is a token which keeps a list of objects containing the token. For the current object r, we use its textual signature $\textsc{Sig}_T(r)$ to probe the corresponding inverted lists. All the objects in the lists will be taken as candidates.

Obviously, these two methods are ineffective since they only use a single constraint to prune candidates.

(3) Spatial and Textual Separately: We build inverted index for spatial and textual signatures separately (Figure 3(a)). For the current object r, we first generate its spatial signature $\textsc{Sig}_S(r)$ and textual signature $\textsc{Sig}_T(r)$. For each grid $g \in \textsc{Sig}_S(r)$, we use it to probe the corresponding spatial inverted index and add all the objects in these lists to spatial candidate set C_S. Meanwhile, for each token $t \in \textsc{Sig}_T(r)$, we scan the corresponding textual inverted list and add the objects to textual candidate set C_T. Then the intersection of C_S and C_T will be taken as the candidates.

Example 4: Take Figure 1 as an example. We maintain inverted index for all the objects that have been visited. Suppose $\tau_s = 0.6$, $\tau_t = 0.7$ and all the tokens have been sorted based on the weight. In Figure 3, we have scanned objects $r_1 \sim r_9$ and the current object is $r_{10} = \{R_{10}; (t_1, t_2, t_5, t_6)\}$. Based on the prefix filter, $\textsc{Sig}_S(r_{10}) = \{g_2, g_3, g_4, g_6, g_7\}$ and $\textsc{Sig}_T(r_{10}) = \{t_1, t_2\}$. We build separate inverted indexes for spatial signatures (grids) and textual signatures (tokens). For object r_{10}, we first use grids in $\textsc{Sig}_S(r_{10})$ to probe the spatial inverted index. Take the first grid g_2 as an example, $\{R_7, R_8\}$ will be added to C_S. After processing all the grids, $C_S = \{R_4, R_5, R_6, R_7, R_8\}$. Then we use tokens in $\textsc{Sig}_T(r_{10})$ to probe the textual inverted lists. Take the first token t_1 as an example, $\{R_7, R_8, R_1, R_3, R_4, R_5\}$ will be added to C_T. After processing all tokens, we have C_T =

$\{R_1, R_3, R_4, R_5, R_6, R_7, R_8\}$. *We then intersect C_T and C_S and get the final candidate set $C(r_{10}) = \{R_4, R_5, R_6, R_7, R_8\}$.*

Notice that in this method, the process of spatial and textual filtering is independent. That is, when we use the separate indexes to generate candidates, we do not consider the pruning power of the other constraint. Based on this observation, we propose the following methods to combine spatial and textual signatures.

(4) First Spatial Then Textual: We can build a two-layer index. If the spatial index is arranged at the top layer, the index is called first-spatial-then-textual (Figure 3(*b*)). Given an object r, we first generate its spatial signature $\text{SIG}_S(r)$ and textual signature $\text{SIG}_T(r)$. The top layer is the grids in $\text{SIG}_S(r)$ and the bottom layer is the inverted lists for tokens in $\text{SIG}_T(r)$. To find the candidates of r, for each grid g in $\text{SIG}_S(r)$ and each token t in $\text{SIG}_T(r)$, if t is in the inverted index of g, the objects in the corresponding inverted lists are candidates. Similarly we can update the index for r.

Example 5: Figure 3(b) shows an example. We first use grids in $\text{SIG}_S(r_{10})$ to find the corresponding inverted index. Take g_2 as an example. The inverted index of g_2 contains the token lists of $\{t_1, t_2, t_3\}$. Then we use tokens in $\text{SIG}_T(r_{10})$ to probe these inverted lists and $\{R_7, R_8\}$ are added into the candidate set. Using the same method, we process all the grids and the candidate set C is $\{R_4, R_5, R_6, R_7, R_8\}$.

(5) First Textual Then Spatial: If the textual index is arranged at the top layer, the index is called first-textual-then-spatial. In Figure 3(*c*), we illustrate the algorithm. Given an object r, we first generate its spatial signature $\text{SIG}_S(r)$ and textual signatures $\text{SIG}_T(r)$. For each token t in $\text{SIG}_T(r)$ and grid g in $\text{SIG}_S(r)$, if g is in the inverted index of token t, the objects in the corresponding inverted list are candidates. Similarly we can update the index for object r.

(6) Hybrid (Spatial and Textual Simultaneously): We will discuss the details in Section 5.

3.3 Sorting Strategies

In our framework, there are still two pivotal problems: (1) how to determine the orders of grids and tokens; and (2) how to sort the objects within each inverted list.

The Order of Grids: The goal of generating signatures is to prune objects as many as possible. Thus we need to choose those grids which can best distinguish one object from others. There are three possible ways: (1) GRID-IDF: A widely accepted method is to sort the grids based on their idf in descending order, where idf is the inverse value of objects that have overlap with the grid. (2) GRID-TOKEN: Though some grids contain large amounts of objects, they may only have a few distinct tokens which increase pruning power. Thus, to further explore the pruning power of tokens, we may sort the grids in the ascending order of the token cardinality within each grid. (3) GRID-AREA: For each object r, the larger area a grid is covered by $\mathcal{M}_r$, the more accuracy the grid represents for $\mathcal{M}_r$. Thus, to better estimate the region, we sort the grids in the descending order of the average overlap and delete those grids with small overlap.

The Order of Tokens: (1) TOKEN-IDF: A general way is to sort the tokens based on idf of tokens, where idf of a token is the inverse value of objects that contain the token. (2) TOKEN-GRID: To further utilize the spatial information of tokens, we can count up the grids each token appears in and delete those tokens appearing in too many grids.

The Order of Objects: If we sort all the objects in order, then only part of objects within the inverted lists need to be scanned. Several strategies can be considered: (1) OBJECT-LENGTH: Similar to the traditional prefix filtering, all the objects are sorted according to token size. Given an object r with $|\mathcal{T}_r|$ tokens, we only need to scan the objects whose token size is between $(\tau_t \times |\mathcal{T}_r|, \frac{|\mathcal{T}_r|}{\tau_t})$. (2) OBJECT-S: We extend this idea to support spatial constraint by sorting objects according to MBR area. Given an object r with region $\mathcal{M}_r$, we only scan the objects whose MBR size is between $(\tau_s \times |\mathcal{M}_r|, \frac{|\mathcal{M}_r|}{\tau_s})$. (3) OBJECT-X(Y): We utilize the plane sweep algorithm in Section 2.2 and sort objects according to x(y)-coordinate. Only the pairs have intersection along at least one axis will be taken as candidates.

3.4 Complexity

We first analyze the space and time complexity and then give some explanations. For constraint of space, we only show the complexity of algorithm "First Spatial Then Textual " as an example. The analysis of others is similar.

Let $|\mathcal{S}|$ denote the size of object set $\mathcal{S}$. We first analyze the space complexity. We divide the entire space into $\mathcal{G}$ grids and build inverted index for the tokens falling in each grid. For an object s, each token in $\text{SIG}_T(s)$ repeatedly appear in the inverted index of $|\text{SIG}_S(s)|$ grids. Suppose the space of each object is M. Then the space complexity for storing objects within inverted lists is $O(M \times \sum_{s\in\mathcal{S}} (|\text{SIG}_S(s)| \times |\text{SIG}_T(s)|))$. Besides, suppose the space of a pointer is P and each grid contains at most $\mathcal{T}_{max}$ distinct tokens. Thus, the space complexity of mapping grids to their corresponding inverted indexes is $O(\mathcal{G} \times P)$. For each inverted index in a grid, the space complexity of mapping the tokens to their corresponding inverted lists is $O(\mathcal{T}_{max} \times P)$. Thus, the overall time space complexity is $O(\mathcal{G} \times P + \mathcal{G} \times \mathcal{T}_{max} \times P + M \times \sum_{s\in\mathcal{S}} (|\text{SIG}_S(s)| \times |\text{SIG}_T(s)|))$. The time complexity contains three parts: sorting, filtering and refining. We mainly focus on the filtering stage. For each object s, we first find the inverted index for each grid in $\text{SIG}_S(s)$ and then use tokens in $\text{SIG}_T(s)$ to probe these inverted indexes. In the worst case, all the objects in the inverted lists will be scanned and added to the candidate set. Thus, the time complexity is $O(\sum_{s\in\mathcal{S}} \sum_{g\in\text{SIG}_S(s)} (O(1) + \sum_{t\in\text{SIG}_T(s)} (O(1) + |\mathcal{I}_{gt}|)))$ where $\mathcal{I}_{gt}$ is the length of inverted list of token t in grid g.

Observe that most time is wasted on scanning inverted lists. Thus, we can optimize the algorithm in two aspects: (1) Strengthen the pruning techniques to avoid visiting unnecessary inverted lists and (2) Reduce the effective length of each inverted list to decrease the scanning objects. For the first aspect, we explore an MBR-Prefix signature in Section 4. For the second aspect, we devise a hybrid signature and discuss how to optimize it in Section 5.

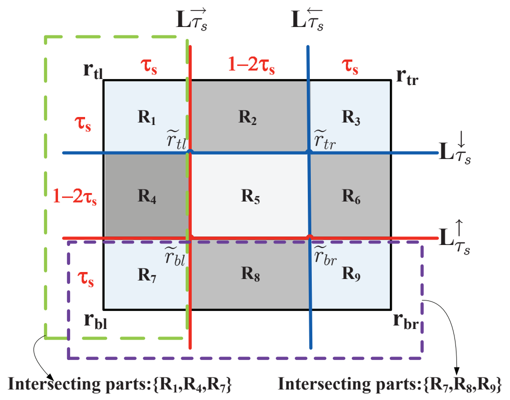

$L_{\tau_s}^{\rightarrow}$
$L_{\tau_s}^{\leftarrow}$
$L_{\tau_s}^{\downarrow}$
$L_{\tau_s}^{\uparrow}$
r_{tl}
r_{tr}
r_{bl}
r_{br}
τ_s
$1-2\tau_s$
R_1 R_2 R_3 R_4 R_5 R_6 R_7 R_8 R_9
$\widetilde{r}_{tl}$ $\widetilde{r}_{tr}$ $\widetilde{r}_{bl}$ $\widetilde{r}_{br}$
Intersecting parts:{R_1,R_4,R_7}
Intersecting parts:{R_7,R_8,R_9}

Fig. 4. MBR-Prefix filtering technique: $\tau_s \le 0.5$

4 MBR-PREFIX BASED FILTER

In this section, we first propose an effective spatial filtering technique in Section 4.1 and then discuss how to incorporate it into filtering algorithms in Section 4.2.

4.1 MBR-Prefix

The grid based spatial filter has a limitation that the filtering power relies largely on the grid granularity. For example, in Figure 1, if we partition the space into 4×4 grids, though r_6 and r_7 have no overlap, they are still taken as candidate since they share common grid g_6. Instead, if we partition the space into 8×8 grids, r_6 and r_7 will be pruned since they appear at different grids. However, as we increase the grid granularity, each signature contains more grids so that the index size will become very large. To address this problem, we propose an MBR-Prefix based filtering technique. The basic idea is to use more accurate spatial information to generate signature. Inspired from the prefix filtering technique [6], we only need to keep specific subregion of $\mathcal{M}_r$ as its spatial signature. The chosen subregion should satisfy the following property: *all the objects similar with r should have overlap with this subregion.* To illustrate the idea clearly, we first introduce some concepts: MBR-Prefix, Representative MBR-Prefix and Minimum MBR-Prefix.

Definition 4: (MBR-PREFIX, REPRESENTATIVE MBR-PREFIX and MINIMUM MBR-PREFIX) Given an object r, any subregion of $\mathcal{M}_r$ is called an *MBR-Prefix* of r. An MBR-Prefix $\mathcal{M}_p$ is called a *representative MBR-Prefix* of r, if for any object s which satisfies $S_{Jac}(r, s) > \tau_s$, we have $\mathcal{M}_p \cap \mathcal{M}_s \neq \phi$. The representative MBR-Prefix with the minimum size is called the *Minimum MBR-Prefix*.

Thus, our goal is to find the minimum MBR-Prefix which can best represent the MBR. Tao et al. [23] proposed a PCR technique (probabilistically constrained region) to process range queries on multi-dimensional uncertain data. They devise effective rules to select an adaptive region for the uncertain data. Inspired from this idea, we develop new pruning techniques for our problem and prove that the region selected by our method is the optimal one. The comparison between PCR and MBR-Prefix is shown in Section 4.2. Now we discuss how to generate the minimum MBR-Prefix in two cases: $\tau_s \le 0.5$ and $\tau_s > 0.5$.

Case 1 - $\tau_s \le 0.5$: Consider the MBR of object r, denoted by $\mathcal{M}(r_{bl}, r_{tr})$, in Figure 4. Its projection along $x(y)$ axis is denoted by *width*(*height*). Let Line(r_{bl}, r_{tl}) denote the left line of the MBR, i.e., the line from the bottom-left point to the top-left point. Let $L_{\tau_s}^{\rightarrow}$ and $L_{\tau_s}^{\leftarrow}$ respectively denote the parallel lines of the left line and the right line with distance $\tau_s \times width$. Let $L_{\tau_s}^{\downarrow}$ and $L_{\tau_s}^{\uparrow}$ respectively denote the parallel lines of the top line and the bottom line with distance $\tau_s \times height$. These four lines generate four intersections $\widetilde{r}_{bl}$, $\widetilde{r}_{tl}$, $\widetilde{r}_{tr}$, $\widetilde{r}_{br}$, and divide $\mathcal{M}_r$ into nine regions ($R_1 \sim R_9$) as illustrated in Figure 4. We omit *width* and *height* in the figure for concise. Notice that the size of $\mathcal{M}_{(R_1 \cup R_4 \cup R_7)}$, the region on the left of $L_{\tau_s}^{\rightarrow}$ is $\tau_s \times |\mathcal{M}_r|$. For any object s which is similar to r, we have $|\mathcal{M}_s \cap \mathcal{M}_r| > \tau_s \times |\mathcal{M}_r|$. Thus, there must be at least one point of s falling into the area on the right of Line $L_{\tau_s}^{\rightarrow}$, *i.e.*, $\mathcal{M}_r \setminus \mathcal{M}_{(R_1 \cup R_4 \cup R_7)}$ (otherwise the intersecting part of r and s can not be larger than $\tau_s \times |\mathcal{M}_r|$). Thus, $\mathcal{M}_r \setminus \mathcal{M}_{(R_1 \cup R_4 \cup R_7)}$ is a representative MBR-Prefix of r. Similarly, $\mathcal{M}_r \setminus \mathcal{M}_{(R_1 \cup R_2 \cup R_3)}$, $\mathcal{M}_r \setminus \mathcal{M}_{(R_3 \cup R_6 \cup R_9)}$ and $\mathcal{M}_r \setminus \mathcal{M}_{(R_7 \cup R_8 \cup R_9)}$ are all representative MBR-Prefixes. We now prove that their intersection area, i.e., $\mathcal{M}_{R_5}$, is the minimum MBR-Prefix of r.

Lemma 2: Given an object r, if $\tau_s \le 0.5$ then $\mathcal{M}_{R_5}$ is the minimum MBR-Prefix of r.

Proof: First we prove $\mathcal{M}_{R_5}$ is a representative MBR-Prefix of r by contradiction. Suppose $\mathcal{M}_{R_5} \cap \mathcal{M}_s = \phi$, then no points of $\mathcal{M}_s$ fall into R_5. To ensure intersection, it must fall into regions $\{R_1, R_3, R_7, R_9\}$ or $\{R_2, R_4, R_6, R_8\}$,we only need to discuss these two cases respectively. If any point of $\mathcal{M}_s$ appears in R_7, to ensure having no overlap with R_5, it at most covers $\mathcal{M}_{R_1 \cup R_4 \cup R_7}$ or $\mathcal{M}_{R_7 \cup R_8 \cup R_9}$. In either case the sum of these areas is no larger than $\tau_s \times |\mathcal{M}_r|$. Thus, $|\mathcal{M}_r \cap \mathcal{M}_s| \le \tau_s \times |\mathcal{M}_r|$ which contradicts with the condition. We can get similar conclusion if any point of s falls into R_4. Thus, $\mathcal{M}_{R_5}$ is a representative MBR-Prefix of r. Then we prove $\mathcal{M}_{R_5}$ is the minimum MBR-Prefix. Suppose there is another representative MBR-Prefix $\mathcal{M}_{R_5'}$ which is smaller than $\mathcal{M}_{R_5}$. Obviously, $\mathcal{M}_{R_5 \setminus R_5'} \neq \phi$ or $\mathcal{M}_{R_5'}$ cannot be smaller than $\mathcal{M}_{R_5}$. However, no matter how small $\mathcal{M}_{R_5 \setminus R_5'}$ is, we can always find similar object which intersects with this area but not intersect with $\mathcal{M}_{R_5'}$. That contradicts with the assumption that $\mathcal{M}_{R_5'}$ is a representative MBR-Prefix. Thus, R_5 is the minimum MBR-Prefix of r. □

Thus, for each object r, we can use its minimum MBR-prefix $\mathcal{M}_{R_5}$ as a spatial signature instead of keeping the entire $\mathcal{M}_r$. More importantly, only those objects intersecting with $\mathcal{M}_{R_5}$ can be similar to r. Notice that when $\tau_s = 0.5$, $\mathcal{M}_{R_5}$ turns into a point (denoted by p). All the objects without intersection with p can be pruned.

Case 2 - $\tau_s > 0.5$: Notice that when $\tau_s > 0.5$, line $L_{\tau_s}^{\rightarrow}$ moves to the right side of line $L_{\tau_s}^{\leftarrow}$ as shown in Figure 5. Like the former case, we can also use the center of R_5 to represent the whole area and all the objects covering this point will be taken as candidates. However, this bound is not tight. For example, consider the MBR-Prefix $\mathcal{M}_{R_2 \cup R_3 \cup R_5 \cup R_6}$. Though it covers point p, it cannot be a candidate since $|\mathcal{M}_{R_2 \cup R_3 \cup R_5 \cup R_6}| = \tau_s^2 \cdot |\mathcal{M}_r| \le \tau_s \cdot |\mathcal{M}_r|$. To

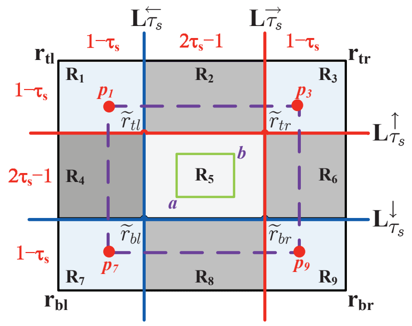

$L_{\overleftarrow{\tau_s}}$
$L_{\overrightarrow{\tau_s}}$
$L_{\overset{\uparrow}{\tau_s}}$
$L_{\overset{\downarrow}{\tau_s}}$
$1-\tau_s$
$2\tau_s-1$
r_{tl}
r_{tr}
r_{bl}
r_{br}
R_1 R_2 R_3 R_4 R_5 R_6 R_7 R_8 R_9
p_1 p_3 p_7 p_9
$\widetilde{r}_{tl}$ $\widetilde{r}_{tr}$ $\widetilde{r}_{bl}$ $\widetilde{r}_{br}$
a
b

Fig. 5. MBR-Prefix filtering technique: $\tau_s > 0.5$

this end, we propose new techniques for $\tau_s > 0.5$.

Similar to the proof of Lemma 2, when $\tau_s > 0.5$, we can prove that $\mathcal{M}_{R_1}$, $\mathcal{M}_{R_3}$, $\mathcal{M}_{R_7}$ and $\mathcal{M}_{R_9}$ are all representative MBR-Prefixes of r. Then s must intersect with regions R_1, R_3, R_7 and R_9 of r simultaneously. Suppose p_1,p_3,p_7,p_9 are four points of s falling in R_1, R_3, R_7 and R_9 as shown in Figure 5. Obviously, $\mathcal{M}(p_7, p_3) \subseteq \mathcal{M}_s$ since p_7 and p_3 are two inner points of s. Thus, *we have an intuition that $\mathcal{M}_s$ must cover some subregions of r*. We can use this property to improve the MBR-Prefix. To illustrate our idea more clearly, we introduce some new concepts, called Coverage MBR-Prefix and Maximum-Coverage MBR-Prefix.

Definition 5: (COVERAGE MBR-PREFIX AND MAXIMUM-COVERAGE MBR-PREFIX) Given an object r, an MBR-Prefix $\mathcal{M}_p$ of r is called a Coverage MBR-Prefix of r, if for any object s satisfying $SJac(r, s) > \tau_s$, we have $\mathcal{M}_p \subseteq \mathcal{M}_s$. Among all these Coverage MBR-Prefixes, we call the largest one as the Maximum-Coverage MBR-Prefix.

Lemma 3: Given an object r, if $\tau_s > 0.5$, $\mathcal{M}_{R_5}$ is the Maximum-Coverage MBR-Prefix of r.

Proof: First we prove $\mathcal{M}_{R_5}$ is a coverage MBR-Prefix of r. As shown in Figure 5, for any object s which is similar to r, we can always find a region $\mathcal{M}(p_7, p_3)$ which satisfies $\mathcal{M}_{R_5} \subseteq \mathcal{M}(p_7, p_3) \subseteq \mathcal{M}_s$. Thus, $\mathcal{M}_{R_5}$ is a coverage MBR-Prefix of r. We then prove that $\mathcal{M}_{R_5}$ is a maximum coverage MBR-Prefix. Suppose there is a coverage MBR-Prefix $\mathcal{M}_{R'_5}$ which is larger than $\mathcal{M}_{R_5}$. Obviously, $\mathcal{M}_{R_5' \setminus R_5} \neq \phi$ or $\mathcal{M}_{R'_5}$ cannot be larger than $\mathcal{M}_{R_5}$. No matter how small $\mathcal{M}_{R_5' \setminus R_5}$ is, we can always find similar object s which does not intersect with this area. Thus, $\mathcal{M}_{R'_5}$ cannot be totally covered by s and $\mathcal{M}_{R_5}$ is the Maximum-Coverage MBR-Prefix of r. □

4.2 MBR-Prefix based Algorithms

We discuss how to utilize MBR-prefixes to generate high quality signatures and incorporate them into our algorithms.

Recall the grid-prefix based spatial signature in Section 3. For each r, we use all the grids which have overlap with r as its signature (i.e., all grids in $\mathcal{G}_r$). Based on MBR-prefix, we can largely reduce the size of these signatures as follows.

Case 1: $\tau_s \leq 0.5$. Based on Lemma 2, given an object r, only the objects which have overlap with r's minimum MBR-prefix can be candidates. To this end, we use the grids that have overlap with r's minimum MBR-prefix $\mathcal{M}_{R_5}$ as r's signatures. Let $\mathcal{G}^p_r$ denote the set of all such grids. Obviously $\mathcal{G}^p_r \subseteq \mathcal{G}_r$. Take $r_{10} = \{R_{10}; [(19, 8), (39, 23)]\}$ as an example. We have $\mathcal{G}_{r_{10}} = \{g_2, g_3, g_4, g_6, g_7, g_8, g_{10}, g_{11}, g_{12}\}$ (Figure 1). Suppose $\tau_s = 0.4$. The $\mathcal{M}_{R_5}$ of r_{10} is $\{[(27, 14), (31, 17)]\}$. Thus, we map $\mathcal{M}_{R_5}$ to grids and get r_{10}'s spatial signature $\mathcal{G}^p_{r_{10}} = \{g_7, g_8\}$. This method is very effective since it reduces an object's entire area to $(1-2\tau_s)^2$ and eliminate objects falling out of this region. For example, if $\tau_s = 0.3$, the area is reduced to 16%; if $\tau_s = 0.4$, the area is sharply decreased to 4%. When $\tau_s = 0.5$, $\mathcal{M}_{R_5}$ turns to a point.

Next we discuss how to incorporate MBR-Prefixes into our algorithm. For simplicity, we take the separated index based method as an example and our method can be easily extended to support other indexes. In this case, we build spatial inverted index based on $\mathcal{G}^p_r$ as follows. The entries are grids in $\mathcal{G}^p_r$. For each grid, its inverted list is composed of the MBR-Prefixes whose signature contains the grid. Consider the current object r. For any object s that has been visited, if r is similar to s, based on the definition of MBR-Prefix, r must have overlap with at least one grid in $\mathcal{G}^p_s$, i.e., $\mathcal{G}_r \cap \mathcal{G}^p_s \neq \phi$. Thus we can take the objects in the inverted list of each grid in $\mathcal{G}_r$ as candidates in terms of spatial constraints. After finding all the candidates of r, we insert its MBR-Prefix into the inverted lists of $\mathcal{G}^p_r$. Notice that by utilizing the plane sweep idea, we do not need to scan the entire inverted list. We sort all the objects s according to Line($\widetilde{s}_{bl}$,$\widetilde{s}_{tl}$) in advance. When coming an object r, only the objects between Line(r_{bl},r_{tl}) and Line($\widetilde{r}_{br}$,$\widetilde{r}_{tr}$) will be taken as candidates within each inverted lists. Besides, we check if $\mathcal{M}_r$ ($\mathcal{M}_s$) intersects with $\mathcal{M}_{R_5}$ of $\mathcal{M}_s$ ($\mathcal{M}_r$).

Case 2: $\tau_s > 0.5$. According to Lemma 3, only the objects entirely covering $\mathcal{M}_{R_5}$ can be candidates. That is, the pivotal points $\widetilde{r}_{tl}, \widetilde{r}_{tr}, \widetilde{r}_{bl}, \widetilde{r}_{br}$ should be covered simultaneously. Notice that these four points are actually determined by two x-coordinates and two y-coordinates since adjacent points have the same x-coordinate or y-coordinate. If we utilize the order of coordinates while building index, then only two points are needed for locating $\mathcal{M}_{R_5}$. For example, we first sort all the objects according to the x-coordinate of the right line. Consider the current object r, for any object s that has been visited, we can prove that *if and only if $\mathcal{M}_r$ covers Line($\widetilde{s}_{bl}$, $\widetilde{s}_{tl}$), r can totally cover the $\mathcal{M}_{R_5}$ of s*. To this end, we take the grids that have overlap with Line($\widetilde{s}_{bl}$, $\widetilde{s}_{tl}$) as the spatial signature of s. Take r_{10} as an example. Suppose $\tau_s = 0.8$. According to Figure 5, $\mathcal{M}_{R_5} = \{(23, 11), (35, 20)\}$. Then we have Line($(\widetilde{r_{10}})_{bl}$, $(\widetilde{r_{10}})_{tl}$) = {(23,11),(23,35)} and $\mathcal{G}^p_{r_{10}} = \{g_7\}$. Notice that though the grids are decreased a lot, when Line($\widetilde{s}_{bl}$, $\widetilde{s}_{tl}$) is long, $\mathcal{G}^p_s$ may still contain too many grids. To further reduce the signature size, we relax the grid selection condition as follows. For grids in $\mathcal{G}^p_s$, we select the grid which has the shortest inverted list in the current inverted index as its signature. (for the first-spatial-then-textual method, we use the sum of length of all inverted lists under the grid). If objects covers Line($\widetilde{s}_{bl}$, $\widetilde{s}_{tl}$), they must have intersection with a grid in $\mathcal{G}^p_s$. Obviously in this case, s has only one grid as its signature.

To incorporate it in our algorithm, we also take the separated index based method as an example. In this case, for the spatial inverted index, the entry is the signature and the inverted list is composed of the MBR-Prefixes whose signature has the grid. Consider the current object r. For any object s that have been visited, according to the analysis above, if r is similar to s, r must cover the signature of s, that is $\mathcal{G}_r$ contains $\mathcal{G}_s^p$. Thus we can take the objects in the inverted list of each grid in $\mathcal{G}_r$ as candidates in terms of spatial constraints. Notice that we do not need to scan the entire inverted list. Based on the definition of maximum coverage MBR-prefix, r must totally cover $\mathcal{M}_{r_5}$ of s, that is, $\widetilde{r}_{tr}.x < s_{tr}.x \le r_{tr}.x$. Thus, for each inverted list, we only need to scan the objects between Line$(\widetilde{r}_{br}, \widetilde{r}_{tr})$ and Line(r_{br}, r_{tr}). After finding all the candidates of r, we need insert its MBR-Prefix into the inverted lists of $\mathcal{G}_r^p$.

Notice that we only utilize the maximum-coverage property of s. Actually this property also works on r, that is, $\mathcal{M}_s$ should totally cover $\mathcal{M}_{R_5}$ of r. For example, if $\widetilde{s}_{bl}$ appears in the bottom-right part of $\mathcal{M}_r$, $|Line(\widetilde{s}_{bl}, \widetilde{s}_{br})|$ is too small so that $|Line(s_{bl}, s_{br})|$ is also small. Then $\mathcal{M}_s$ cannot totally cover $\mathcal{M}_{R_5}$ of r and can be pruned. Based on this observation, we propose a tighter grid selection to further reduce the number of visiting lists. Suppose the width of r is w. If $\tau_s > 0.5$, we can find a point $r'_{tr} = (r_{tr}.x - w \times \tau_s^2, r_{tr}.y)$ and we only need to visit the inverted lists corresponding to the grids of region $\mathcal{M}(r_{bl}, r'_{tr})$. The correctness is formalized in Lemma 4. For constraint of space, we omit the proof.

Lemma 4 (Signature Selection): Given an object s with width w, when $\tau_s > 0.5$, we can find a point $r'_{tr} = (r_{tr}.x - w \times \tau_s^2, r_{tr}.y)$ and we only need to use grids of $\mathcal{M}(r_{bl}, r'_{tr})$ to visit inverted index.

Comparison between MBR-Prefix and PCR [23]: Although both of the two methods utilize spatial (or probability) thresholds to prune some unnecessary subregions, they have the following differences. (1) The problem definitions (and similarity functions used) are different. PCR focuses on region queries on uncertain data while we emphasize on spatio-textual data. PCR is a search problem while ours is a join problem. (2) We prove that the subregion selected by our method is the optimal one for our problem. However, for uncertain data it is rather hard to select the optimal one and PCR gives a heuristic method. (3) Different from PCR, we use Jaccard as our similarity function and propose new pruning techniques as discussed in Section 3.3 and Section 4.2. (4) Besides prefix-based pruning techniques, we seamlessly integrate the MBR-Prefix filter into our algorithm to support efficient spatio-textual similarity joins.

5 HYBRID SIGNATURE

In this section, we discuss how to seamlessly integrate the spatial pruning and textual pruning to improve the join performance. We first propose a tokenMBR based hybrid signature which can automatically select grid granularity for each token in Section 5.1. We then discuss how to support R-S join and other similarity functions in Section 5.2.

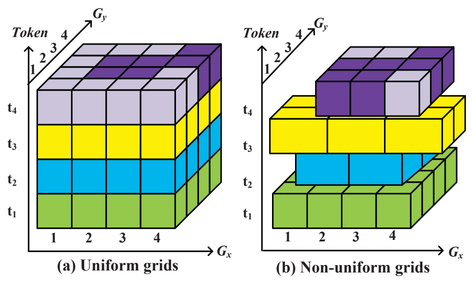


Fig. 6. A hybrid signature pattern

5.1 TokenMBR based Hybrid Signatures

To integrate the spatial and textual signatures of an object, a natural idea is to do a cartesian product on its token prefix and grid prefix. For example, in Figure 6(a), each cell in the cube is uniquely identified by the tuple (g, t) where g is the grid ID in the partition and t is the token ID. Take object r_2 as an example, using the MBR-Prefix technique in Section 4, its token prefix is $\{t_4\}$. Its spatial region $\mathcal{M}_{r_2}$ has overlap with grid $\{g_{12}, g_{16}\}$. Thus, by using cartesian product, the hybrid signature of r_2, denoted by $\textsc{Sig}_H(r_2)$, will be $\{(g_{12}, t_4), (g_{16}, t_4)\}$. However this method has two weaknesses. First, it involves large numbers of signatures. Second, it partitions the entire space into grids identically (Figure 6(a)) and does not consider the token distribution. We propose two effective techniques to improve the signature based method. Before we introduce the technical details, we first define a concept, tokenMBR[28].

Definition 6 (tokenMBR): Given a token t, the tokenMBR of t is defined as the minimum bounding rectangle of all the objects that contain t, denoted by $\mathcal{M}_t$.

First, consider the frequent tokens which are contained by many objects (e.g., Mcdonald's and Gas Stations). It is better to use fine-grained grids to distinguish such objects. We will give an experimental discussion under different grid granularity in Section 5.2.

Second, consider those infrequent tokens which are contained by few objects (e.g., Statue of Liberty and Central Park which only appear in New York). If we partition them using the same granularity as those frequent ones, the corresponding inverted lists only has few objects, which actually wastes the pruning power within each inverted list. Obviously, we can integrate some infrequent tokens to reduce the signature size.

Based on these two observations, we devise a tokenMBR based hybrid signature as shown in Figure 6(b). It can automatically select grid granularity for each token. The basic idea is: (1) For each token, we explore different partition granularity according to its minimum bounding region and frequency. We define a threshold τ_n to restrict the number of objects falling in each grid. Suppose the frequency of token t is f_t. Then we evenly partition its tokenMBR into f_t/τ_n parts. (2) For infrequent tokens, we further combine them and partition them into several groups. We utilize the same token to represent the tokens in the same group using a mapping table, in order to reduce the signature size. We also expect that combining infrequent tokens will not decrease the pruning power.

To this end, we propose a combination principle to combine infrequent tokens. Consider two objects in a candidate pair whose spatial component and textual component are both dissimilar. It is unnecessary to utilize both of the constraints to prune this candidate. In other words, even if we only use one of the constraint (i.e., MBR prefix), they can still be pruned. Based on this idea, we give the token combination principle: For all tokens whose tokenMBRs have no overlap along x-axis, we can combine them into one group and utilize a single token to replace the tokens in the group. More importantly, our token combination based method will not involve more candidates as formalized in Lemma 5.

Lemma 5: For all tokens whose tokenMBRs have no overlap along x-axis, if we combine these tokens, the token combination based method will not involve more candidates.

Proof: Suppose $\mathcal{M}_{t_1}$ and $\mathcal{M}_{t_2}$ are the tokenMBRs of token t_1 and t_2 respectively. $\mathcal{S}_{t_1}$ and $\mathcal{S}_{t_2}$ are the object sets containing t_1 and t_2. Suppose $\mathcal{M}_{t_1}$ and $\mathcal{M}_{t_2}$ have no overlap along x-axis and $(\mathcal{M}_{t_1})_{tr}.x \le (\mathcal{M}_{t_2})_{bl}.x$. Thus, for any object pair (r, s) satisfying $r \in \mathcal{S}_{t_1}$ and $s \in \mathcal{S}_{t_2}$, we have $(\mathcal{M}_r)_{tr}.x \le (\mathcal{M}_{t_1})_{tr}.x \le (\mathcal{M}_{t_2})_{bl}.x \le (\mathcal{M}_s)_{bl}.x$. Notice that in our MBR-Prefix based algorithms, all the objects are sorted according to x-axis in advance. The objects within each inverted list are also in order of x-axis and only those pairs intersecting along x-axis will be visited. Therefore, even if we add r and s to the same inverted list, they have no overlap along x-axis and will not be regarded as candidate. □

Utilizing this principle, we model the token combination problem into the token partition problem. Next we formally define this problem: Given n tokens $t_1, t_2, \ldots, t_n$, each token t_i is associated with an interval $\mathcal{I}_i$ (e.g., the width of its tokenMBR), and we want to divide the n tokens into m non-intersecting groups $G_1, G_2, \ldots, G_m$. The cost of each group G_i is the interval overlaps of objects in the group, which is defined as

$$C_i = \frac{\sum_{\forall t_j, \forall t_k \in G_i} |\mathcal{I}_j \cap \mathcal{I}_k|}{|\bigcup_{t_j \in G_i} \mathcal{I}_j|}$$

where $\sum_{\forall t_j, \forall t_k \in G_i} |\mathcal{I}_j \cap \mathcal{I}_k|$ is the total overlap of any two intervals in group G_i and $|\bigcup_{t_j \in G_i} I_j|$ is the maximum spanning of the intervals in G_i. Our goal is to minimize the sum of cost, i.e., $\sum_{i=1}^{n} C_i$.

Lemma 6: The token partition problem is NP-complete.

Proof: We prove that the token partition problem can be converted from the weighted set-cover problem which has been proven to be a NPC problem. The weighted set cover problem is defined as: Given a universe of elements $\mathcal{U} = \{1, 2, \ldots, n\}$ and a family of subsets $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_m\}$. Each subset $\mathcal{S}_i$ is associated with a weight w_i. We need to find a cover $C \subseteq \mathcal{S}$ such that $\bigcup_{s \in C} s = \mathcal{U}$ and $\sum_{s \in C} w(s)$ is minimized. In our problem, each token t_i corresponds to the element and each group G_i corresponds to the subset. The cost C equals to the weight of subset. And our goal is to minimize the sum of cost which is equal to minimizing the sum of subset weights. Thus the token partition problem is NP-complete. □

Since the token partition problem is NP-complete, we introduce an approximate algorithm. At each step, for each token, the algorithm selects the current best group and adds it into the group. We maintain a priority queue sorted by the costs in an ascending order. At the beginning, for each of the m groups, we initiated it with cost 0 and push it into the priority queue. Inspired from the plane sweep algorithm, we first sort the n tokens according to the left coordinate of their intervals increasingly. Then we scan these tokens sequentially. For each coming token t, we pop the group with the minimum cost from the priority queue, denoted by G_{Min}. We insert t to G_{Min} and update its cost by adding the new overlaps. Then we push G_{Min} back to the priority queue. After processing all the tokens, we get m non-intersecting groups. For the tokens within the same group, we integrate their frequencies and tokenMBRs for further partition. Meanwhile, we maintain a table to lookup the mapping relation from a combined token to a token group.

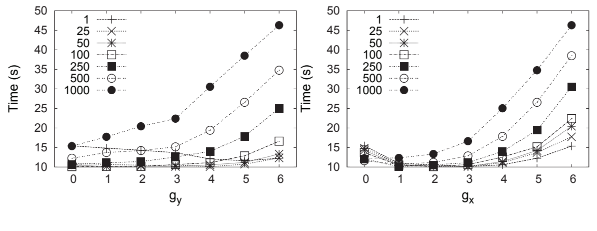
Next we discuss how to combine the token partition based idea into the hybrid signature.

The tokenMBR based signature: We sort all the tokens according to their frequencies. We pick the last 10% as the infrequent tokens. For those tokens, we combine them into several groups using the token partition algorithm above and maintain a mapping table. Given an object r, for each token t in $\textsc{Sig}_T(r)$, we lookup its corresponding token t' using the mapping table. According to its frequency, we partition the tokenMBR $\mathcal{M}_{t'}$ into grids and calculate $\textsc{Sig}_S(r)$ under this partition. Then we combine t' with each grid in $\textsc{Sig}_S(r)$ to generate the signature.

Extensions to our algorithm: Similar to the methods in Section 4.2, the entry is a signature and the inverted list is composed of the MBR Prefixes that have the signature. Consider the current object r. For any object s that has been visited, if r is similar to s, $\mathcal{G}_s^p$ and $\mathcal{G}_r$ must have common grid, and $\mathcal{T}_r$ and $\mathcal{T}_s$ must have common token, i.e., $\textsc{Sig}_H^p(s) \cap \textsc{Sig}_H(r) \neq \phi$. Then the objects in the inverted lists of $\textsc{Sig}_H(r)$ will be taken as candidates. In the update step, we add MBR Prefix of r to the corresponding inverted lists of $\textsc{Sig}_H^p(r)$.

5.2 Discussions

The Effect of Grid Granularity Selection: Figure 7 shows the performance of hybrid signature under different partition granularity. We fix $\mathcal{G}_x$ (or $\mathcal{G}_y$) and then vary $\mathcal{G}_y$ (or $\mathcal{G}_x$) from 1 to 1000. We have two observations: (1) With the increase of $\mathcal{G}_x$ (or $\mathcal{G}_y$), the time curve first falls and then rises again. (2) In a curve with a larger $\mathcal{G}_x$ (or $\mathcal{G}_y$), the lowest point appears at a small $\mathcal{G}_y$ (or $\mathcal{G}_x$). The reason is twofold: (1) If we partition the space coarsely, each object contains fewer grids. The objects within each inverted list increases. Lots of time is wasted on scanning the inverted list innerly. (2) If we partition the space finely, each object contains more grids and the signature size increases accordingly. According to the algorithm,



(a) Twitter (b) Twitter

Fig. 7. Evaluation on grid granularity

we should probe the inverted index for each grid-token pair. Then lots of time is wasted on repeatedly probing the inverted index. Thus the grid granularity problem is converted to making a tradeoff between the length of each inverted list and the number of entries of the inverted index. Thus, in Section 5.1, we propose a hybrid method which can automatically select grid granularity for each token. Experimental results in Section 6.4 show that this strategy achieves good performance.

Supporting Two Different Datasets $\mathcal{R} \neq \mathcal{S}$: Our algorithm can be easily extended to the case that $\mathcal{R} \neq \mathcal{S}$. First, we fix a global order for objects in $\mathcal{R} \cup \mathcal{S}$. Second, we scan these objects sequentially. For each coming object, we use its signature to probe the inverted lists and skip those prefixMBRs belonging to the same set. We then update the index with its signature. Finally we refine the candidates.

Supporting Other Similarity Functions: Besides Jaccard, our algorithm can also support other similarity functions. Consider objects $r(s)$ with MBR $\mathcal{M}_r$ ($\mathcal{M}_s$) and token set $\mathcal{T}_r$ ($\mathcal{T}_s$). We define their spatial and textual similarity under different metrics as Table 1. To present clearly, we define function $\mathcal{W}(r,s) = \sum_{t \in \mathcal{T}_s \cap \mathcal{T}_r} w(t)$ and $\mathcal{W}(r) = \sum_{t \in \mathcal{T}_r} w(t)$. We can extend our algorithms to support "Dice" and "Cosine" by deducing $|\mathcal{M}_r \cap \mathcal{M}_s|$ and $\mathcal{W}(r,s)$ from these similarity functions.

TABLE 1
Other similarity metrics

Functions	Spatial similarity	Textual similarity
Dice	$S_{Dice}(r,s) = \frac{2 \cdot \lvert\mathcal{M}_r \cap \mathcal{M}_s\rvert}{\lvert\mathcal{M}_r\rvert + \lvert\mathcal{M}_s\rvert}$	$T_{Dice}(r,s) = \frac{2\mathcal{W}(r,s)}{\mathcal{W}(r)+\mathcal{W}(s)}$
Cosine	$S_{Cos}(r,s) = \frac{\lvert\mathcal{M}_r \cap \mathcal{M}_s\rvert}{\sqrt{\lvert\mathcal{M}_r\rvert \cdot \lvert\mathcal{M}_s\rvert}}$	$T_{Cos}(r,s) = \frac{\mathcal{W}(r,s)}{\sqrt{\mathcal{W}(r) \cdot \mathcal{W}(s)}}$

Dice similarity:

$$S_{Dice}(r,s) > \tau_s \Rightarrow |\mathcal{M}_r \cap \mathcal{M}_s| > \frac{\tau_s}{2-\tau_s} \times |\mathcal{M}_r|$$

$$T_{Dice}(r,s) > \tau_t \Rightarrow \mathcal{W}(r,s) > \frac{\tau_t}{2-\tau_t} \times \mathcal{W}(r)$$

Cosine similarity:

$$S_{Cos}(r,s) > \tau_s \Rightarrow |\mathcal{M}_r \cap \mathcal{M}_s| > \tau_s^2 \times |\mathcal{M}_r|$$

$$T_{Cos}(r,s) > \tau_t \Rightarrow \mathcal{W}(r,s) > \tau_t^2 \times \mathcal{W}(r)$$

6 Experimental Study

We conducted extensive experiments on two datasets to evaluate our proposed techniques. The results show that our methods outperform other solutions and achieve high performance.

TABLE 2
Dataset statistics.

Property	USA	Twitter
Object number (Million)	10	10
Number of keywords	615,407	1,238,389
Average token length	9.64	8.48
Data size (GB)	1.14	0.801

6.1 Experimental Settings

Datasets: We use two datasets: USA and Twitter. Table 2 summarizes these two datasets. The Twitter dataset is a real dataset. We crawled 10 million tweets with region and textual information from Twitter[5]. The USA dataset is a synthetic dataset which combines the Points of Interest(POIs) in US and the publications in PUBMED. The US contains 11 million POIs and the PUBMED contains 20 million publications. To extend POIs to MBRs, we took these points as the center of objects and randomly generated the width and height following a power law distribution. We then combined these MBRs and publications randomly.

Experimental Environment: All the algorithms were implemented in C++ and run on a Linux machine with an Intel(R) Xeon(R) CPU X5670 @ 2.93GHz and 48GB memory. The algorithms were complied using GCC 4.2.3.

Parameter Setting: Unless stated explicitly, parameters were set as follows by default: $\tau_t = \tau_s = 0.8$, $\mathcal{G}_x = \mathcal{G}_y = 100$, $\tau_n = 1000$. We sorted objects by HighX and sorted grids and tokens by IDF.

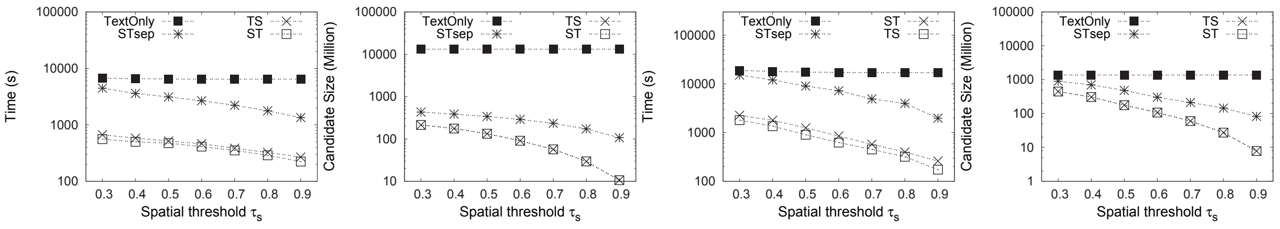
6.2 Evaluating Different Signature Schemes

We evaluate five signature schemes, textual only (`TextOnly`), spatial only (`SpaOnly`), spatial and textual separate (`STsep`), first-spatial-then-textual (`ST`), first-textual-then-spatial (`TS`) in Section 3 by varying τ_s and τ_t. Since `SpaOnly` is much slower than other methods, we omit it in the figures.

The effect of τ_s. To evaluate the effect of spatial threshold τ_s, we fixed τ_t to 0.8 and compared the time and candidate size. Figure 8 shows the results. We can see that `TextOnly` has the worst performance and is almost a straight line since no spatial constraint is used. Though `STsep` is 1.5-8 times faster than `TextOnly`, it is still 5-10 times slower than `ST`. The reason is that `ST` uses spatial pruning first and then textual pruning to get the candidates, while `STsep` takes these two constraints as totally independent filters. Notice `ST` and `TS` almost have the same performance since they both use two filters. In Figure 8(b) and 8(d), we can see that `ST` and `TS` always have the same candidate size. Thus, the organizing sequence of spatial and textual signatures will not effect the final candidates, but will effect the time in pruning them.

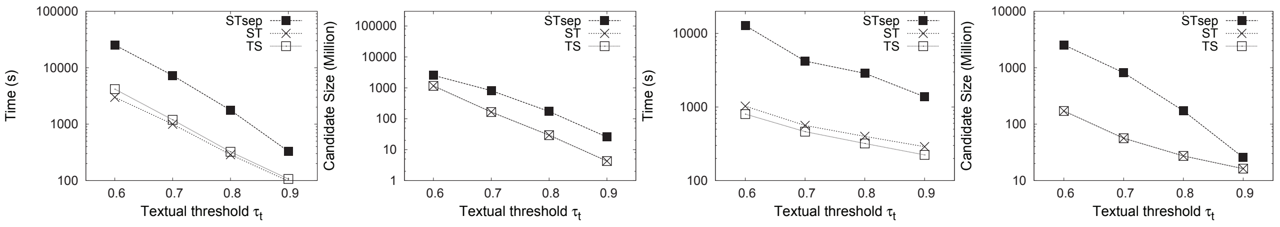
The effect of τ_t. We fixed τ_s to 0.8 to evaluate the effect under different textual thresholds. Figure 9 shows the results. Since `SpaOnly` and `TextOnly` are much slower than others, we omit them in the figure. We can see that `ST` and `TS` greatly outperform `STsep` in both time and

5. http://twitter.com



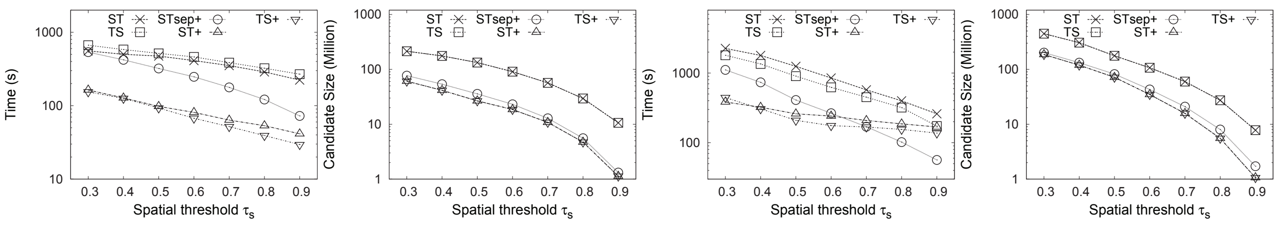
(a) USA-Time (b) USA-Candidate (c) Twitter-Time (d) Twitter-Candidate

Fig. 8. Evaluation on different signature schemes by varying τ_s



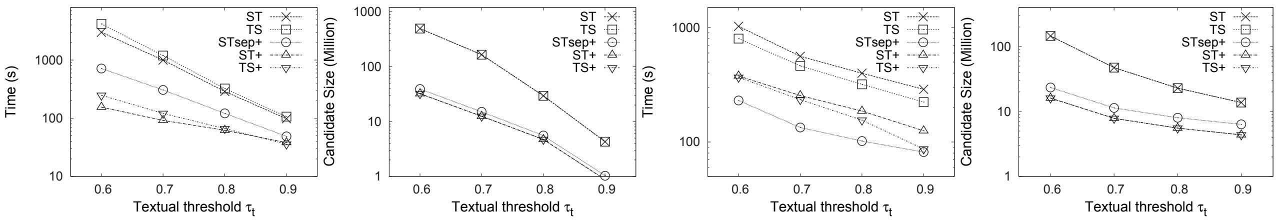
(a) USA-Time (b) USA-Candidate (c) Twitter-Time (d) Twitter-Candidate

Fig. 9. Evaluation on different signature schemes by varying τ_t



(a) USA-Time (b) USA-Candidate (c) Twitter-Time (d) Twitter-Candidate

Fig. 10. MBR-Prefix vs Non-MBR-Prefix by varying τ_s



(a) USA-Time (b) USA-Candidate (c) Twitter-Time (d) Twitter-Candidate

Fig. 11. MBR-Prefix vs Non-MBR-Prefix by varying τ_t

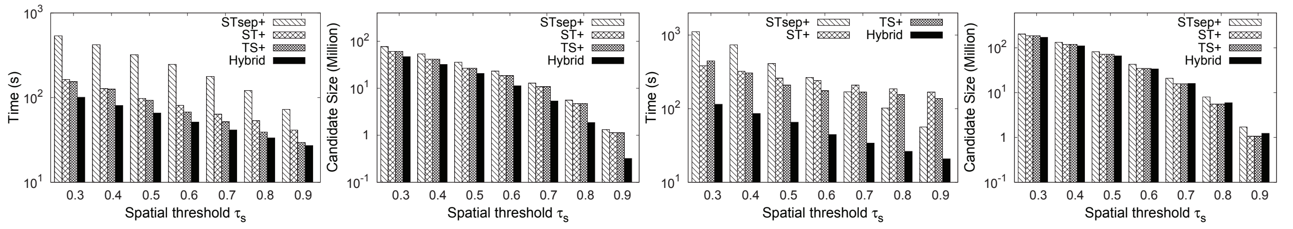
candidate size. For example, in Twitter, when τ_t = 0.8, STsep took 2871 seconds while TS only took 319 seconds.

6.3 MBR-Prefix vs Non-MBR-Prefix

We evaluate MBR-Prefix based filtering techniques in Section 4. In the figures, the algorithms with "+" denote the improved algorithms by incorporating MBR-Prefix. For example, TS$^+$ denotes the algorithm of incorporating MBR-Prefix into the TS algorithm.

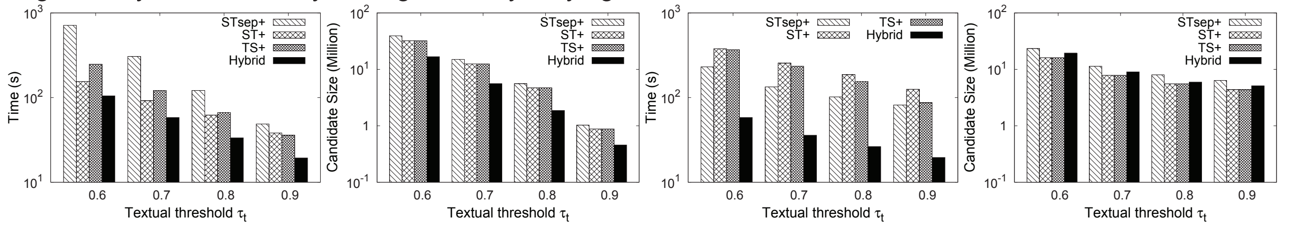
The effect of τ_s. We fixed τ_t to 0.8 and the results are shown in Figure 10. We omit STsep since it is far more slower than others. We can see that the MBR-Prefix technique significantly improves the performance of ST and TS. Take ST as an example. In USA, when τ_s = 0.6, ST took 411 seconds, while ST$^+$ only took 81 seconds. In (b), ST had 90.7 million candidates while ST$^+$ had only 18.5 million candidates. There are two main reasons. (1) We use the MBR-Prefix to substitute the entire region. The size of each grid signature is reduced and thus some inverted lists can be pruned. (2) By utilizing the plane sweep idea within each inverted list, only those MBR-Prefixes falling in specific regions will be taken as candidates. Notice that the time gap between STsep$^+$ and ST$^+$ becomes smaller when τ_s increases. For example, in Twitter, ST$^+$ outperform STsep$^+$ at the beginning. When τ_s increases to 0.7, STsep becomes slower than STsep$^+$. This is because when τ_s is close to 1, the number of objects pruned by the spatial constraint decreases sharply and the filtering time of STsep$^+$ decreases accordingly. However, compared to STsep$^+$, ST$^+$ took much more time in maintaining the index and reducing redundant candidates which may cause extra time in the pruning stage. Thus, when τ_s is large, the performance of STsep$^+$ may be better than ST$^+$ and TS$^+$.

The effect of τ_t. The result is shown in Figure 11. ST$^+$



(a) USA-Time (b) USA-Candidate (c) Twitter-Time (d) Twitter-Candidate

Fig. 12. Hybrid vs Non-Hybrid signature by varying τ_s



(a) USA-Time (b) USA-Candidate (c) Twitter-Time (d) Twitter-Candidate

Fig. 13. Hybrid vs Non-Hybrid signature by varying τ_t

and $\texttt{TS}^+$ again outperform ST and TS. Notice that in USA, $\texttt{STsep}^+$ is 1.3-4 slower than $\texttt{ST}^+$ and $\texttt{TS}^+$. However, in Twitter, $\texttt{STsep}^+$ has the best performance among all the methods. The reason is similar to the above-mentioned discussion. We can see that MBR-Prefix based algorithm is effective in pruning candidates.

6.4 Hybrid vs Non-Hybrid

To evaluate the performance of Hybrid, we compared it with our best three methods $\texttt{ST}^+$, $\texttt{TS}^+$ and $\texttt{STsep}^+$. Figure 12 and 13 show the results under different τ_s and τ_t respectively. Take Figure 12 as an example. Hybrid is 1.5-8 times faster than $\texttt{ST}^+$ and 3-10 times faster than $\texttt{STsep}^+$. For example, in Twitter, when $\tau_s = 0.6$, $\texttt{STsep}^+$ took 265 seconds and $\texttt{ST}^+$ took 240 seconds, while Hybrid only took 45 seconds. There are two main reasons. (1) Hybrid tries to find the best grid granularity for every token. It partitions each token using its tokenMBR and adjusts the grid granularity according to the token frequency. Instead, $\texttt{ST}^+$ partitions all the tokens according to the same grid granularity. (2) Hybrid devises different grid orders for different tokens. For each token t, its grids are sorted according to the GRID-IDF of all the objects containing t. However in $\texttt{ST}^+$, all the objects have the same grid order. In Figure 12(d), the candidate size of $\texttt{ST}^+$ and Hybrid were 5.54 and 5.95 million respectively. Although the candidate size of Hybrid increased slightly due to the combination of infrequent tokens, we can see that Hybrid still achieves best performance due to reducing signatures.

6.5 Evaluating Different Sorting Strategies

To evaluate the performance of different sorting strategies (Section 3.3), we compared the time of methods ST, TS and STsep under different sorting strategies by varying τ_s ranging from 0.5 to 0.9. Experiments show that ST and TS always have the same performance. Thus we only show the results of ST and STsep for constraint of space.

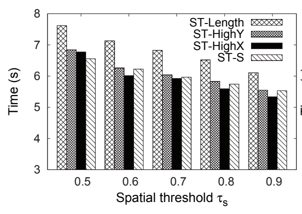
The effect of different object order. We use "-Length", "-HighX", "-HighY" and "-S" to denote the sorting strategies of Object-Length, Object-X, Object-Y and Object-S respectively. As shown in Figure 14, sorting by token length is slower than others. The reason may be in Twitter, each tweet is limited to 140 words so that lots of tweets have almost the same length. Thus the length pruning power is weakened a lot. Besides, sorting by HighX is slightly faster than HighY which means that the objects distribute almost uniformly along x-axis and y-axis.

The effect of different grid order. We use "-Area", "-IDF" and "-Token" to denote the strategies sorting by Grid-area, Grid-IDF, and Grid-token separately. As shown in Figure 15, sorting by IDF and Token almost have the same performance. That is because objects in Twitter almost have the same length. Thus the objects and tokens contained by one grid have the similar distribution. Besides, sorting by area is 20-100 times slower than others. Though objects have different distribution in different grids, the sum of their overlaps within a grid is possibly similar.

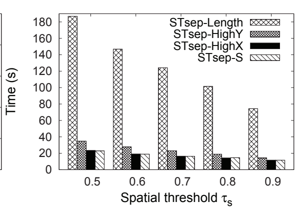
The effect of different token order. Sorting by IDF and grids almost have the same performance. The reason is similar to above. Thus, we omit this figure.

6.6 Scalability

In this section, we evaluate the time and index scalability of our methods. Take Twitter as an example. We vary the object size from 1-10 million. In Figure 16(a), we can see that STsep, $\texttt{ST}^+$, $\texttt{TS}^+$ and Hybrid almost achieve a linear scalability. In Figure 16(b), we compared the memory size of different methods. To present their differences clearly, we did not include the memory size of loading data. We can see that ST and TS took the most memory since they only use one constraint to generate signature. By using MBR-prefix technique, the index size of $\texttt{STsep}^+$ is only slightly larger than $\texttt{ST}^+$ and $\texttt{TS}^+$. Hybrid outperforms others greatly by using the minimum time and memory. The reason is that we combine the inverted lists for those infrequent tokens. Thus we can avoid repeatedly probing some unnecessary objects since the entries of inverted lists are reduced.

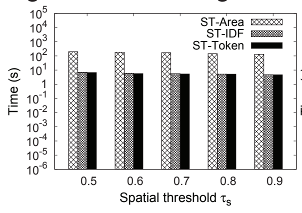


(a) Twitter

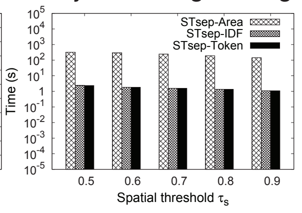


(b) Twitter

Fig. 14. Evaluating different object sorting strategies

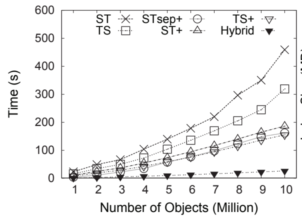


(a) Twitter

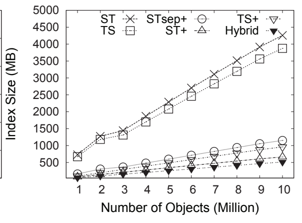


(b) Twitter

Fig. 15. Evaluating different grid sorting strategies



(a) Twitter



(b) Index size

Fig. 16. Evaluation on time and index scalability

6.7 Comparison with Existing Methods

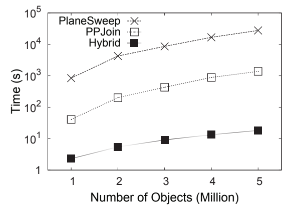
In this section, we compared our best algorithm `Hybrid` with two state-of-the-art methods: Plane Sweep [21] and a prefix filter based method `PPJoin` [26]. We vary object size from 1-5 million and evaluate the time and candidate size. The result is shown in Figure 17. We can see that `Hybrid` largely outperforms `PlaneSweep` and `PPJoin` in both time and candidate size. `PlaneSweep` has the worst performance since it only uses the intervals along X(Y)-axis to pruning candidates. For example, in Figure 17(a), when object size is 3 million, `PlaneSweep` took 8627 seconds and `PPJoin` took 431 seconds. But `Hybrid` only took 9 seconds.

6.8 Evaluating $\mathcal{R}$-$\mathcal{S}$ Join

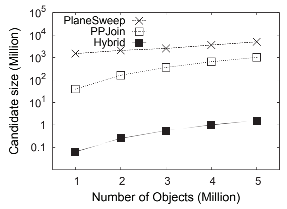
To evaluate the performance of our algorithm on different datasets, we evenly divide Twitter into two parts, denoted by $\mathcal{R}$ and $\mathcal{S}$. We fixed the object size of $\mathcal{R}$ (5 million) and varied the objects size of $\mathcal{S}$ from 1-5 million. We compare the elapsed time of `Hybrid` under different thresholds. Figure 18(a) shows the result. We can see that our hybrid method also achieves good scalability on $\mathcal{R}$-$\mathcal{S}$ join. Take the case $\tau_s = \tau_t = 0.7$ as an example. It took 47 seconds when the object size is 2 million. When the object size is 5 million, it only took 97 seconds. Thus, our algorithm can support the $\mathcal{R}$-$\mathcal{S}$ join effectively.

6.9 Evaluating Different Similarity Functions

We fixed τ_s and τ_t to 0.8 and evaluate the performance under different similarity functions ("Jaccard", "Dice", "Co-

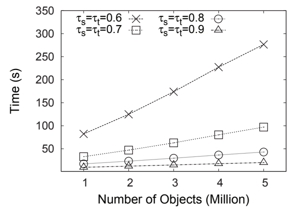


(a) Twitter

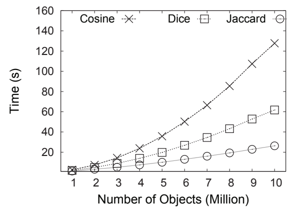


(b) Candidates

Fig. 17. Comparison with existing methods



(a) $\mathcal{R} \neq \mathcal{S}$



(b) Similarity functions

Fig. 18. Evaluating $\mathcal{R}$-$\mathcal{S}$ join and similarity functions

sine"). The result is shown in Figure 18(b). Our algorithm also scales well under different similarity functions. Notice that these three functions had different performance under the same τ_s and τ_t. The reason is that when "Dice" and "Cosine" are converted to "Jaccard", their thresholds are decreased compared to "Jaccard".

7 Related Work

To the best of our knowledge, there is no study of others on the spatio-textual similarity join problem. Here we review some related work on spatial join, string similarity join, and spatial keyword search.

Spatial Join: Many methods have been proposed to study the spatial join problem [4], [14], [15], [17], [18], [20]. Plane sweep [4], [21] is a classic method discussed.To index spatial objects, existing work [4], [19] mainly used R-tree like structure [11] to organize data in a hierarchically way. Based on the structure, some work [4], [13] studied different probing strategies such as depth-first and breadth-first. Other work [18], [20] used hash methods by partitioning the space into grids. Lo et al. [17] proposed a seeded tree to combine the two structures. Jacox et al. [14] gave a survey about existing spatial join techniques. In addition, some works focused on specific predicates. Hjaltason et al [12] studied the distance join problem and Yiu et al. [27] addressed the common influence join problem. Brinkhoff et al. [3] focused on processing irregular objects.

String Similarity Join: Recently there are many studies on string similarity joins [1], [2], [6], [10], [22], [25].[2] proposed the prefix filtering technique to support similarity join with Jaccard and Cosine functions. Xiao et al [25] extended this method to support the edit-distance function. Wang et al [24] proposed to use trie structure to support similarity joins. In this paper we used the prefix filtering framework to support similarity join on spatio-textual data. Obviously our problem is different from theirs.

Spatial Keyword Search: There are many studies on spatial keyword search recently [5], [7], [8], [9], [29]. They focused on integrating inverted indexes and R-tree to support spatial keyword search, which, given a set of spatio-textual objects and a query, finds the relevant answers to the query. Our join problem is different from their work for two reasons: (1)For the join problem, the index should be built online. But in their work, the index is built in advance and they mainly focus on the time of query processing. (2)Our query is two set of data so that we can utilize their distribution to avoid comparing dissimilar pairs. However, the query for their problem is a single data.

This paper is a major-value added version of our previous work [16]. The significant additions in this extended manuscript are summarized as follows: (1)We propose a hybrid signature which integrates spatial and textual pruning power. We prove the token-partition problem is NP-complete and develop a greedy algorithm. Section 5.1 is newly added. To evaluate its performance, we also conduct new experiments and Section 6.4 is newly added. (2)We extend our algorithm to support R-S join and other similarity functions in Section 5.2. We also conduct new experiments. Section 6.8 and 6.9 are newly added. (3)We discuss different sorting strategies in Section 3.3 and analyze the complexity in Section 3.4. Corresponding experiments are added in Section 6.5. (4)We add the details, running examples, and proofs to our MBR-Prefix based technique in Section 4. (5)We add the comparison of candidate sizes of different algorithms in Sections 6.2 and 6.3. We also add some new experiments in Sections 6.5, 6.6 and 6.7.

8 Conclusion

In this paper, we study a new research problem called spatio-textual similarity join. We devise a prefix-filter based framework and propose several possible solutions. To achieve higher performance, We further develop an MBR-Prefix based filtering technique which can prune large number of dissimilar objects. We also propose a hybrid signature by integrating spatial information and textual descriptions. We model a token partition problem and prove it is NP-complete. We then propose an approximate method and integrate it to our method. Experiments show that our methods can achieve high performance and scale well.

References

[1] A. Arasu, V. Ganti, and R. Kaushik. Efficient exact set-similarity joins. In *VLDB*, pages 918–929, 2006.

[2] R. J. Bayardo, Y. Ma, and R. Srikant. Scaling up all pairs similarity search. In *WWW*, pages 131–140, 2007.

[3] T. Brinkhoff, H.-P. Kriegel, R. Schneider, and B. Seeger. Multi-step processing of spatial joins. In *SIGMOD Conference*, pages 197–208, 1994.

[4] T. Brinkhoff, H.-P. Kriegel, and B. Seeger. Efficient processing of spatial joins using r-trees. In *SIGMOD Conference*, pages 237–246, 1993.

[5] X. Cao, G. Cong, C. S. Jensen, and B. C. Ooi. Collective spatial keyword querying. In *SIGMOD*, pages 373–384, 2011.

[6] S. Chaudhuri, V. Ganti, and R. Kaushik. A primitive operator for similarity joins in data cleaning. In *ICDE*, page 5, 2006.

[7] G. Cong, C. S. Jensen, and D. Wu. Efficient retrieval of the top-k most relevant spatial web objects. *PVLDB*, 2(1):337–348, 2009.

[8] J. Fan, G. Li, L. Zhou, S. Chen, and J. Hu. Seal: Spatio-textual similarity search. *PVLDB*, 5(9):824–835, 2012.

[9] I. D. Felipe, V. Hristidis, and N. Rishe. Keyword search on spatial databases. In *ICDE*, 2008.

[10] L. Gravano, P. G. Ipeirotis, H. V. Jagadish, N. Koudas, S. Muthukrishnan, and D. Srivastava. Approximate string joins in a database (almost) for free. In *VLDB*, pages 491–500, 2001.

[11] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *SIGMOD Conference*, pages 47–57, 1984.

[12] G. R. Hjaltason and H. Samet. Incremental distance join algorithms for spatial databases. In *SIGMOD Conference*, pages 237–248, 1998.

[13] Y.-W. Huang, N. Jing, and E. A. Rundensteiner. Spatial joins using r-trees: Breadth-first traversal with global optimizations. In *VLDB*, pages 396–405, 1997.

[14] E. H. Jacox and H. Samet. Spatial join techniques. *ACM Trans. Database Syst.*, 32(1):7, 2007.

[15] N. Koudas and K. C. Sevcik. Size separation spatial join. In *SIGMOD Conference*, pages 324–335, 1997.

[16] S. Liu, G. Li, and J. Feng. Star-join: spatio-textual similarity join. In *CIKM*, pages 2194–2198, 2012.

[17] M.-L. Lo and C. V. Ravishankar. Spatial joins using seeded trees. In *SIGMOD Conference*, pages 209–220, 1994.

[18] M.-L. Lo and C. V. Ravishankar. Spatial hash-joins. In *SIGMOD Conference*, pages 247–258, 1996.

[19] N. Mamoulis and D. Papadias. Integration of spatial join algorithms for processing multiple inputs. In *SIGMOD Conference*, pages 1–12, 1999.

[20] J. M. Patel and D. J. DeWitt. Partition based spatial-merge join. In *SIGMOD Conference*, pages 259–270, 1996.

[21] F. Preparata and M. Shamos. *Computational geometry: an introduction*. Texts and monographs in computer science. Springer-Verlag, 1985.

[22] S. Sarawagi and A. Kirpal. Efficient set joins on similarity predicates. In *SIGMOD*, pages 743–754, 2004.

[23] Y. Tao, R. Cheng, X. Xiao, W. K. Ngai, B. Kao, and S. Prabhakar. Indexing multi-dimensional uncertain data with arbitrary probability density functions. In *VLDB*, pages 922–933, 2005.

[24] J. Wang, G. Li, and J. Feng. Trie-join: Efficient trie-based string similarity joins with edit-distance constraints. *PVLDB*, 3(1):1219–1230, 2010.

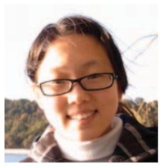
[25] C. Xiao, W. Wang, and X. Lin. Ed-join: an efficient algorithm for similarity joins with edit distance constraints. *PVLDB*, 1(1):933–944, 2008.

[26] C. Xiao, W. Wang, X. Lin, and J. X. Yu. Efficient similarity joins for near duplicate detection. In *WWW*, 2008.

[27] M. L. Yiu, N. Mamoulis, and P. Karras. Common influence join: A natural join operation for spatial pointsets. In *ICDE*, pages 100–109, 2008.

[28] D. Zhang, Y. M. Chee, A. Mondal, A. K. H. Tung, and M. Kitsuregawa. Keyword search in spatial databases: Towards searching by document. In *ICDE*, pages 688–699, 2009.

[29] D. Zhang, B. C. Ooi, and A. K. H. Tung. Locating mapped resources in web 2.0. In *ICDE*, pages 521–532, 2010.



Sitong Liu is currently a PhD candidate in the Department of Computer Science, Tsinghua University, Beijing, China. Her research interests mainly include data integration, spatio-textual data query, and spatial database.

Guoliang Li received his PhD degree in Computer Science from Tsinghua University, Beijing, China in 2009. He is currently working as an associate professor in the Department of Computer Science, Tsinghua University, Beijing, China. His research interests mainly include data cleaning and integration, spatial databases, and crowdsourcing.

Jianhua Feng received his B.S., M.S. and PhD degrees in Computer Science from Tsinghua University. He is currently working as a professor of Department Computer Science in Tsinghua University. His main research interests include native XML database, data mining, and keyword search over structure & semi-structure data.