Fast Similarity Search on Keyword-Induced Point Groups

Zhe Li
Dept. of Computing,
Hong Kong Polytechnic Univ.
richie.li@connect.polyu.hk

Yu Li
Dept. of Computer Science and
Technology, Hangzhou Dianzi Univ.
flyzeroliyu@gmail.com

Man Lung Yiu
Dept. of Computing,
Hong Kong Polytechnic Univ.
csmlyiu@comp.polyu.edu.hk

ABSTRACT

Location-based social media (e.g., Twitter, Foursquare) have been generating massive amount of geo-textual data. In this paper, we represent the spatial distribution of a keyword by the group of locations tagged with such keyword. Given a query keyword, our problem is to find k keywords with the most similar distribution of locations. Such query finds applications in targeted marketing and recommendation. The performance of existing solutions degrade when different point groups have significant overlapping, which happens rather frequently in real data. We propose efficient techniques to process similarity search on point groups. Experimental results on Twitter data demonstrate that our solution is faster than the state-of-the-art by up to 6 times.

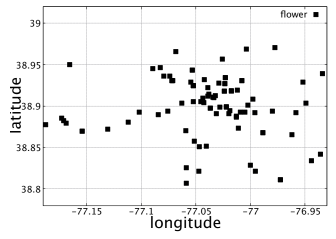
CCS CONCEPTS

• **Information systems → Spatial-temporal systems**; *Geographic information systems*; *Nearest-neighbor search*;

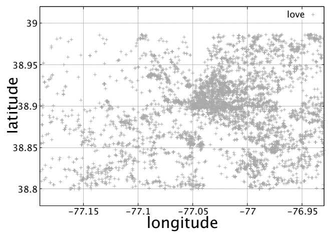
KEYWORDS

Hausdorff distance, Similarity Searching, Spatio-Textual Searching

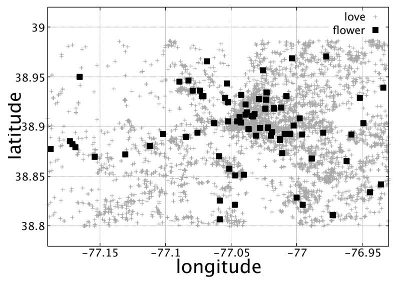
ACM Reference Format:
Zhe Li, Yu Li, and Man Lung Yiu. 2018. Fast Similarity Search on Keyword-Induced Point Groups. In *26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL '18), November 6–9, 2018, Seattle, WA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3274895.3274920



(a) point group of "flower": G_{flower}



(b) point group of "love": G_{love}



(c) comparing two point groups G_{flower} and G_{love}

Figure 1: Keyword-induced point groups in Washington

1 INTRODUCTION

In this era of smart phones, massive amount of geo-textual data are being continuously produced by end-users. For instance, each tweet[1] contains a text message (up to 140 characters) as well as a location. Social photo sharing websites (e.g. Flickr)[2] contain photos with both descriptive tags and locations. Foursquare,[3] a location-based social network, provides the "check-in" function for end-user to share a message tagged with a location.

With the above geo-tagged messages, we propose the concept of *keyword-induced point group*. Given a keyword key, we form a point group G_{key} as the set of locations such that their messages contain the keyword key. To illustrate this concept, we extract all tweets located in Washington, and then visualize the point groups of different keywords. For the keyword "flower", we obtain the point group G_{flower} and then plot its distribution in Figure 1a. Similarly, for the keyword "love", its point group G_{love} is shown in Figure 1b.

We are interested in comparing the spatial distributions between two keyword-induced point groups. By displaying both G_{flower} and G_{love} in the same map (in Figure 1c), we observe that most of the tweets containing "flower" are close to some tweets containing "love". That would reveal certain connection between the keywords "flower" and "love". Such information can be exploited in applications like targeted marketing and recommendation. For instance, a flower shop may wish to show advertisements (e.g., Twitter Ads) to nearby users who have just posted tweets about "love". Alternatively, when a user posts a tweet about "flower", the system may recommend a nearby tweet about "love". According to a survey [5], some works have considered using location information to recommend geo-tagged messages. We are the first to consider *the similarity of location distribution* in tweet recommendation.

In this paper, we study the similarity search problem on keyword-induced point groups. Given a query point group Q, we wish to find K point groups such that they are the most

[1]https://twitter.com/
[2]https://www.flickr.com/
[3]https://foursquare.com/

Table 1: Top-10 search results on point groups in Washington (under different similarity measures)

Q: point group of "flower"

$dist_H(Q,G)$	**love** > u > time > lol > today > life > **good** > day > night > **girl**
$dist_{SH}(Q,G)$	yep > perform > **art** > dj > noticed > easter > chicago > lay > downtown > hide
$dist_{cen}(Q,G)$	jw > nr > npr > dual > puck > reuter > exhibiting > swarmin > stroll > ajc

Q: point group of "president"

$dist_H(Q,G)$	time > love > people > night > day > u > good > **life** > school > **great**
$dist_{SH}(Q,G)$	**pain** > **lose** > send > j > tweeting > set > mile > type > smile > **office**
$dist_{cen}(Q,G)$	place > infinitely > seewhy > cannot > suite > credible > regency > odd > frap > toomuch

similar to Q with respect to a similarity measure. Following the literature [2], we consider both the Hausdorff distance $dist_H(Q,G)$ and the symmetric Hausdorff distance $dist_{SH}(Q,G)$ as distance measures between two point groups Q and G. As a baseline for comparison, we also consider the Euclidean distance between the centroids of point groups $dist_{cen}(Q,G)$. The equations for these distance measures are given below:

Hausdorff distance:	$dist_H(Q,G) = \max_{q_i \in Q} \min_{p_j \in G} dist(q_i, p_j)$
Symmetric Hausdorff distance:	$dist_{SH}(Q,G) = \max\{dist_H(Q,G), dist_H(G,Q)\}$
Euclidean distance between centroids:	$dist_{cen}(Q,G) = dist(q_c, p_c)$

where $dist(q,p)$ denotes the Euclidean distance between two points, and q_c, p_c are the center locations of Q and G, respectively.

To understand the quality of the above distance measures, we conduct a case study on keyword-induced point groups derived from Twitter data located in Washington. We consider two query keywords "flower" and "president", and show their top-10 similarity search results (for each distance measure) in Table 1. The result keywords (in bold) are viewed to be meaningful for human users. Observe that the results found by $dist_{cen}(Q,G)$ are not meaningful. Both $dist_H(Q,G)$ and $dist_{SH}(Q,G)$ can lead to some meaningful results. Thus, in this paper, we take the Hausdorff distance $dist_H(Q,G)$ as the distance measure by default. We will also extend our techniques for the symmetric Hausdorff distance $dist_{SH}(Q,G)$.

The state-of-the-art solution for our problem is [2]. Assume that all data point groups have been indexed by R-trees. At query time, it builds an R-tree for the query point group Q, then utilizes minimum bounding rectangles (MBRs) to derive lower bound distance for $dist_H(Q,G)$ and attempt pruning unpromising data point groups. Nevertheless, the solution in [2] has not taken the characteristics of keyword-induced point groups into account. Figure 2 demonstrates two keyword-induced point groups in Washington, again extracted from Twitter data. G_{kid} denotes the point group for the keyword "kid" and $G_{president}$ represents the point group for the keyword "president". Observe that regions covered by two point groups overlap heavily, thus rendering MBR-based lower bound distances loose.

In this paper, we take the above characteristics into careful consideration when designing our techniques. First, we propose a representative-based lower bound for pruning. The idea is to extract a representative subset $Q' \subseteq Q$ such that its distribution is similar to Q, and then compute $dist_H(Q',G)$ as the lower bound distance of $dist_H(Q,G)$. We present a greedy approach to select a representative subset Q' that yields tighter lower bound. Another merit is that the above lower bound supports incremental computation. Even if the pruning is not successful, the computation effort can be recycled in computing the exact distance $dist_H(Q,G)$. Furthermore, we design a technique to filter out unpromising data point groups early in order to achieve further speedup. Experimental results on Twitter data demonstrate that our solution is faster than the state-of-the-art by up to 6 times.

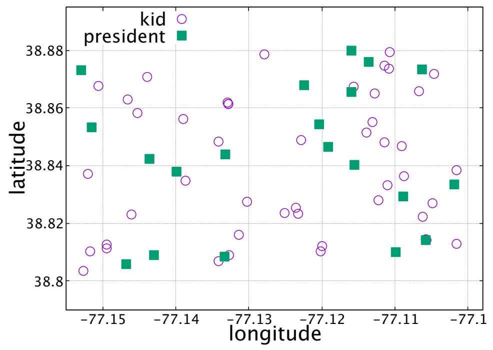


Figure 2: Two point groups in Washington

The remaining of this paper is organized as follows. Section 2 defines our problem formally. Section 3 presents our proposed techniques. Section 4 evaluates the efficiency of our techniques on Twitter data. Section 5 reviews related work. Finally we conclude with future research directions in Section 6.

2 PROBLEM STATEMENT

We are given a raw dataset $\mathcal{D}_{raw}$ of messages, where each message is associated with a location (i.e., point) and a set of keywords. An example of the raw dataset is shown in Table 2. We ignore other attributes (e.g., user ID, timestamp) in this paper.

Based on the raw dataset, we define the concept of *keyword-induced point group* as follows.

Definition 1 (Keyword-induced point group). *Given a keyword key, the point group G_{key} is defined as the set of all locations (in the raw dataset $\mathcal{D}_{raw}$) that share the same keyword.*

For example, the keyword "park" corresponds to the point group G_{park}, which contains four locations: p_1, p_2, p_3, p_4.

Table 2: Example of raw dataset $\mathcal{D}_{raw}$

Message ID	Location	Keywords	Other attributes
1	p_1	park	⋯
2	p_2	park, animal	⋯
3	p_3	park, lake	⋯
4	p_4	park	⋯
5	p_5	mountain, animal	⋯
6	p_6	mountain, lake	⋯
7	p_7	mountain	⋯

In this paper, we adopt the Hausdorff distance to measure the distance between two point groups.

Definition 2 (Hausdorff distance). *Given two point groups Q and G, the Hausdorff distance from Q to G is defined as:*

$$dist_H(Q,G) = \max_{q_i \in Q} \min_{p_j \in G} dist(q_i, p_j)$$

where $dist(q_i,p_j)$ denotes the Euclidean distance between two points.

As an example, we are given two point groups $Q = \{q_1,q_2,q_3,q_4\}$ and $G = \{p_1,p_2,p_3,p_4\}$ in Figure 3. Each dashed line indicates the minimum distance from a query point $q_i \in Q$ to group G. Thus, the Hausdorff distance $dist_H(Q,G)$ equals to $\max\{2.8, 2.5, 1.4, 1\} = 2.8$.

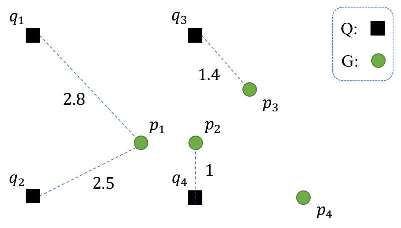


Figure 3: Hausdorff distance $dist_H(Q,G)$

Definition 3 (Similarity search). *Let $\mathcal{D}_{groups}$ be the collection of point groups (derived from the raw dataset $\mathcal{D}_{raw}$). Given a query point group Q and the number K of results, the similarity search problem retrieves point groups $G \in \mathcal{D}_{groups}$ having K smallest $dist_H(Q,G)$.*

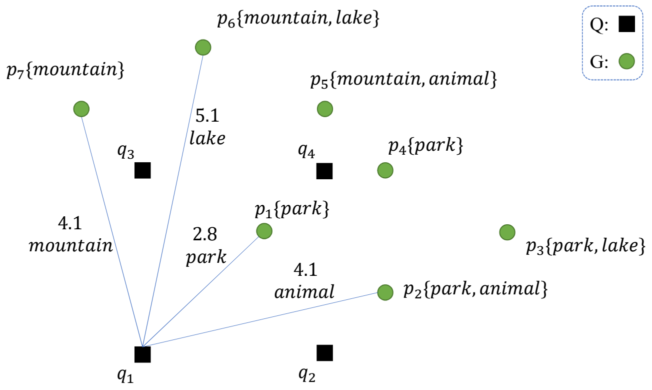
Our problem definition focuses on point groups that are formed by locations of text messages sharing a common keyword. This is also the main difference from the problem definition in [2].

We illustrate an example of similarity search in Figure 4. The dataset $\mathcal{D}_{groups}$ contains four point groups: G_{park}, G_{animal}, $G_{mountain}$, G_{lake}. Each line indicates the Hausdorff distance from the query point group Q to a point group in $\mathcal{D}_{groups}$. Assuming $K = 1$, the result is the point group G_{park}.

Table 3 summarizes the list of symbols used in this paper.

3 OUR SOLUTION

We first present our techniques for bounding the Hausdorff distance. Next we develop a technique to filter candidates. Then we combine our techniques into a similarity search algorithm. Finally, we extend our techniques for the symmetric Hausdorff distance.



$G_{park} = \{p_1,p_2,p_3,p_4\}$, $G_{animal} = \{p_2,p_5\}$,
$G_{mountain} = \{p_5,p_6,p_7\}$, $G_{lake} = \{p_3,p_6\}$

Figure 4: Example of similarity search

Table 3: List of symbols

Symbol	Meaning
q_i	a query point
p_j	a location (i.e., data point)
Q	the query point group
G	a point group
$\mathcal{D}_{groups}$	a collection of point groups
$dist(q_i,p_j)$	Euclidean distance between two points
$dist_H(Q,G)$	Hausdorff distance from Q to G

3.1 Representative-based Lower Bound

The Hausdorff distance $dist_H(Q,G_{key})$ is expensive to compute, incurring $O(|Q| \cdot |G_{key}|)$ time. To skip such expensive computation, we will develop a fast lower bound function $LB(Q,G_{key})$ so that $LB(Q,G_{key}) \le dist_H(Q,G_{key})$. During similarity search, we maintain the threshold $dist_{kBest}$ for the best k Hausdorff distance of point groups examined so far. If a point group satisfies $LB(Q,G_{key}) \ge dist_{kBest}$, then G_{key} can be safely pruned without computing $dist_H(Q,G_{key})$.

Our idea is to pick a representative subset Q' of the query point group Q. Then we propose the following lower bound function:

$$LB_{rep}(Q,G_{key}) = dist_H(Q',G_{key}) \quad (1)$$

To facilitate fast lower bound computation, we introduce a parameter α and require that Q' contains exactly α points. The following lemma establishes the lower bound property of $LB_{rep}(Q,G_{key})$.

Lemma 3.1. *Let Q' be a subset of the query point group Q. For any point group G_{key}, it holds that:*

$$LB_{rep}(Q,G_{key}) \le dist_H(Q,G_{key})$$

Proof. Since $Q = Q' \cup (Q - Q')$, we obtain:

$$dist_H(Q,G_{key}) = \max\{\max_{q_i \in Q'} \min_{p_j \in G} dist(q_i,p_j), \max_{q_i \in (Q-Q')} \min_{p_j \in G} dist(q_i,p_j)\} \quad (2)$$

Thus we get: $dist_H(Q, G_{key}) \geq \max_{q_i \in Q'} \min_{p_j \in G} dist(q_i, p_j)$, then the lemma is proved. □

We demonstrate the pruning power of this lower bound function in Figure 5. We are given two point groups: $Q = \{q_1, q_2, \cdots, q_7\}$ and $G_{key} = \{p_1, p_2, p_3, p_4\}$. Let the representative subset be $Q' = \{q_1, q_2, q_3\}$ and the threshold be $dist_{kBest} = 1.8$. The lower bound $LB_{rep}(Q, G_{key}) = dist_H(Q', G_{key}) = \max\{1.5, 2, 2.1\}$ equals to 2.1. Since $dist_H(Q', G_{key}) > dist_{kBest}$, we prune the point group G_{key} without having to compute the exact $dist_H(Q, G_{key})$.

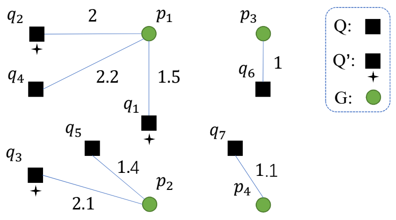


Figure 5: Pruning

The tightness of the lower bound depends on the choice of the subset Q'. Continuing with the example in Figure 5, if we pick the representative subset as $Q' = \{q_1, q_5, q_7\}$, then the lower bound becomes $\max\{1.5, 1.4, 1.1\} = 1.5$, which is looser than before (2.1).

Intuitively, the lower bound can be tightened by selecting a subset Q' whose distribution is similar to that of Q'. We propose a greedy algorithm to achieve this (see Algorithm 1). The parameter α indicates the number of representatives to be chosen. First, we compute the centroid of Q and then select the closest point from Q to the centroid as the first representative. For each query point $q_i \in Q$, we keep track of its distance to the nearest representative RepDist(q_i). The query point with the largest RepDist(q_i) is chosen as the next representative. This procedure is repeated until α representatives have been chosen from Q.

Algorithm 1 Greedy representatives

```
Require: query point group $Q$, number of representatives $\alpha$
 1: $Q' \leftarrow \emptyset$
 2: $q_{rep} \leftarrow$ the point in $Q$ nearest to the centroid of $Q$
 3: append $q_{rep}$ to $Q'$; remove $q_{rep}$ from $Q$
 4: for each point $q_i \in Q$ do
 5:     RepDist($q_i$) $\leftarrow \infty$
 6: while $|Q'| < \alpha$ do
 7:     for each point $q_i \in Q$ do
 8:         RepDist($q_i$) $\leftarrow \min\{dist(q_i, q_{rep}), \text{RepDist}(q_i)\}$
 9:     $q_{rep} \leftarrow$ the point in $Q$ with the lowest RepDist($q_i$)
10:     append $q_{rep}$ to $Q'$; remove $q_{rep}$ from $Q$
11: return $Q'$
```

With the example in Figure 5, we illustrate how the above algorithm works. Assume that $\alpha = 3$. First, the point q_1 is chosen as the first representative. In the second iteration, the point q_2 is chosen because it is the furthest from q_1. In the third iteration, the point q_3 is chosen because it yields the largest distance of $\min\{dist(q_i, q_1), dist(q_i, q_2)\}$.

The time complexity of Algorithm 1 is $O(\alpha \cdot |Q|)$. Observe that the subset Q' is independent of the point group G_{key}. Thus, it suffices to execute Algorithm 1 once, regardless of the number of point groups in the dataset $\mathcal{D}_{groups}$.

3.2 Incremental Lower Bound Computation

The lower bound function $LB_{rep}(Q, G_{key})$ exhibits the decomposable property, namely that:

$$dist_H(Q', G_{key}) = \max\{\ dist_H(Q' - \{q_i\}, G_{key}),\ \min_{p_j \in G_{key}} dist(q_i, p_j)\ \}$$

We exploit this property to compute the lower bound in incremental fashion. To illustrate this idea, we consider the example in Figure 5 and show the computation steps in Table 4. We maintain the running lower bound by the variable $\widehat{LB}$ and initialize it to 0. In each iteration, we compute the term $\min_{p_j \in G_{key}} dist(q_i, p_j)$ for the current query point q_i, then update $\widehat{LB}$ (if $\widehat{LB}$ is smaller), and attempt pruning by comparing $\widehat{LB}$ with the threshold $dist_{kBest}$. For example, given that $dist_{kBest} = 1.8$, we terminate in the second iteration because the lower bound (2) is already higher than $dist_{kBest}$. This technique would further reduce the computation cost of the lower bound function. Notice in this example we have brought representatives (i.e., Q') to the front and keep their original chosen order.

Table 4: Example of incremental lower bound computation

Iteration	Current point	Subset Q'	Lower bound $\widehat{LB}$
1	q_1	$\{q_1\}$	1.5
2	q_2	$\{q_1, q_2\}$	$\max\{1.5, 2\} = 2$
3	q_3	$\{q_1, q_2, q_3\}$	$\max\{2, 2.1\} = 2.1$
...	...	...	...
7	q_7	$\{q_1, q_2, \cdots, q_7\}$	2.2

In addition, even if the pruning is not successful, the computation effort is not wasted. It suffices to examine the remaining points in Q and then update the lower bound in incremental fashion, as shown in Table 4. Upon examining all query points (i.e., reaching iteration 7 in Table 4), the lower bound becomes the exact Hausdorff distance $dist_H(Q, G_{key})$.

We demonstrate the effect of incremental lower bound computation on Twitter data below, using two sample point groups for Q and G respectively. In this test, we compare two methods for obtaining the subset Q': (i) our method for placing representatives of Q into Q', (ii) randomly placing points of Q into Q'. Figure 6 plots the lower bound value of each method with respect to the number of processed query points. Clearly, our method yields much tighter lower bound than random.

3.3 Optimization in Lower Bound Computation

Recall from the previous subsection that we compute the term $\min_{p_j \in G_{key}} dist(q_i, p_j)$ in each iteration, and then update the running lower bound $\widehat{LB}$.

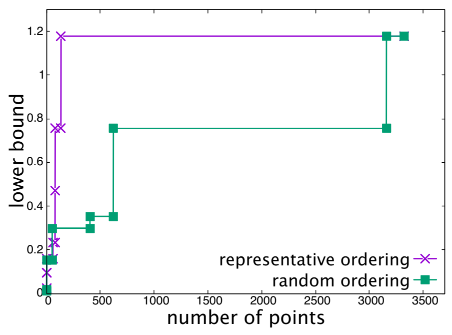


Figure 6: Incremental lower bound on Twitter data

Surprisingly, it is possible to maintain $\widehat{LB}$ correctly without computing the exact value of $\min_{p_j \in G_{key}} dist(q_i, p_j)$. Observe that $\widehat{LB}$ is updated only when $\min_{p_j \in G_{key}} dist(q_i, p_j)$ exceeds $\widehat{LB}$. If there exists a point $p_j \in G_{key}$ such that $dist(q_i, p_j) < \widehat{LB}$, then it means the term $\min_{p_j \in G_{key}} dist(q_i, p_j)$ cannot further increase the value of $\widehat{LB}$.

To fully exploit the above observation, we need to quickly find a point $p_j \in G_{key}$ such that $dist(q_i, p_j)$ is as low as possible, and then test whether $dist(q_i, p_j) < \widehat{LB}$. Our heuristic is:

(1) first find a nearby query point (say, q_{near}) from q_i (among the examined query points),
(2) then retrieve a data point p_j close to q_{near}.

For example, in Figure 7, suppose that we have examined the first query point q_1 and get a nearby point from G_{key} (e.g., point p_1). Upon processing the second query point q_2, we identify a nearby query point (i.e., q_1) and retrieve its corresponding data point (i.e., p_1). Then, we test whether $dist(q_2, p_1) < \widehat{LB}$; if yes, then we skip the computation of $\min_{p_j \in G_{key}} dist(q_2, p_j)$.

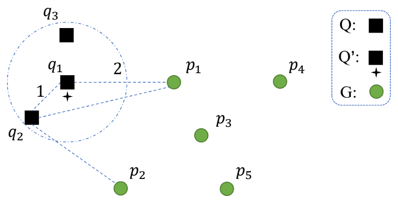


Figure 7: Upper bound

We proceed to discuss efficient implementation of the above technique. In particular, both Steps 1 and 2 can be computed in amortized $O(1)$ time. Step 1 depends only on the points in Q, but not on the points in G_{key}. To support this step, it suffices to precompute a nearby query point for each $q_i \in Q$, before processing any point group in the dataset $\mathcal{D}_{groups}$. Step 2 takes $O(1)$ time because we should have processed the query point q_{near} before considering the current query point q_i.

3.4 Candidate Retrieval

The techniques in previous subsections attempt to evaluate the test $dist_H(Q, G_{key}) \geq dist_{kBest}$ quickly, without invoking the exact Hausdorff distance computation if possible.

In this subsection, we present an orthogonal technique to reduce the number of times to execute the test $dist_H(Q, G_{key}) \geq dist_{kBest}$. The idea is to retrieve candidate point groups (from the dataset $\mathcal{D}_{groups}$) that can potentially enter into the best k query results. This technique consists of three steps:

(1) first sample β ($\geq k$) point groups to obtain the threshold $dist_{kBest}$,
(2) then retrieve candidate point groups,
(3) finally process the candidate point groups to get the final query results.

We call the parameter β to be the sampling ratio. Our experimental study shows that the performance remains stable when β grows larger than 10% of dataset; hence we set β to 10% of the number of point groups by default. Observe that both Steps 1 and 3 can be accelerated by using our techniques proposed in previous subsections.

We elaborate how to retrieve candidate point groups efficiently (Step 2). This step requires a parameter γ, which denotes the number of probing query points. First, we choose γ query points from Q. Then, for each chosen query point q_i, we issue a range search at q_i with radius as $dist_{kBest}$ to retrieve the collection of keywords within the range. Next, we compute the intersection of such collections in order to obtain the candidate keywords (i.e., candidate point groups). We establish the correctness of computing the intersection in the following lemma.

Lemma 3.2. *If $dist_H(Q, G_{key}) \leq dist_{kBest}$, then for every query point $q_i \in Q$ there exists a point $p_j \in G_{key}$ such that $dist(q_i, p_j) \leq dist_{kBest}$.*

An optimization is to remove candidate point groups that have been processed in Step 1.

Figure 8 illustrates an example and Table 5 depicts how our idea works. Suppose that the chosen query points are q_1, q_2. After performing range search at q_1, we obtain four keywords (k_1, k_2, k_3, k_4) as well as their distances to q_1. In case a keyword occurs multiple times (e.g., k_2), it suffices to keep its closest distance to q_1 (i.e., 1.1). Similarly, we execute range search at q_2 and retrieve three keywords (k_2, k_3, k_5). Then, we compute the intersection between these collection of keywords and obtain k_2, k_3. During this process, we obtain lower bound values for the point groups of these keywords for free. For example, the lower bound of k_3 is taken as the maximum distance seen from q_1, q_2, i.e., $\max\{1.8, 1.0\} = 1.8$.

We note that the choice of chosen query points may affect the filtering power of the above step. If the chosen query points are too near, then their range results tend to have significant overlap and thus degrade the filtering power. We recommend to choose such query points based on Algorithm 1, as the first few query points selected in Algorithm 1 tend to be far apart.

3.5 Our Algorithms

In this section, we combine our proposed techniques and present our similarity search algorithm (as Algorithm 2). We denote the

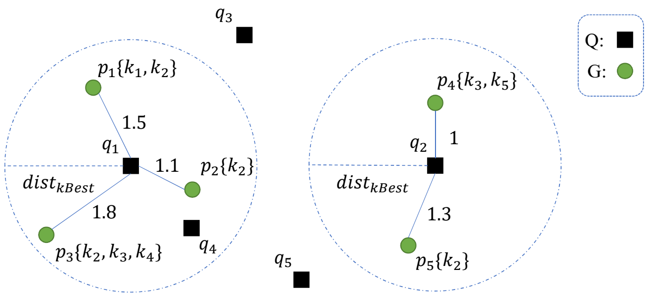


Figure 8: Computing candidate point groups

Table 5: Example of computing candidates

	Candidates
In the range of q_1	$(k_2, 1.1)$ $(k_1, 1.5)$ $(k_3, 1.8)$ $(k_4, 1.8)$
In the range of q_2	$(k_3, 1.0)$ $(k_5, 1.0)$ $(k_2, 1.3)$
Combined list	$(k_2, 1.3)$ $(k_3, 1.8)$

result set by *resultSet* and the best k Hausdorff distances by the threshold $dist_{kBest}$. The techniques applied in statements are indicated clearly in the algorithm. For example, the techniques for representative-based lower bound (cf. Section 3.1) are applied at lines 4 and 10, and candidate retrieval (cf. Section 3.4) is applied at line 5.

The incremental distance computation module (Algorithm 3) is used to compute the Hausdorff distance between two point groups Q and G. It utilizes the threshold $dist_{kBest}$ for early pruning. In Algorithm 2, the above module is used to compute lower bound distance (at line 10) as well as the remaining terms for the exact distance (at line 13).

Regarding the optimization technique in Section 3.3, it is effective only when the data point group G_{key} contains many points. When G_{key} is large (say, containing more than 10 points), we apply the above optimization technique at line 15.

3.6 Extensions

In this section, we consider the symmetric variant of the Hausdorff distance, as defined below.

Definition 4 (Symmetric Hausdorff distance). *The symmetric Hausdorff distance between point groups Q and G_{key} is:*

$$dist_{SH}(Q, G_{key}) = \max\{dist_H(Q, G_{key}), dist_H(G_{key}, Q)\}$$

We illustrate an example for the symmetric Hausdorff distance in Figure 9, which contains two groups: $Q = \{q_1, q_2, q_3, q_4\}$ and $G = \{p_1, p_2, p_3, p_4\}$. Each dashed line indicates the closest distance from $q_i \in Q$ to G, or the closest distance from $p_i \in G$ to Q. Thus, we get $dist_H(Q, G) = \max\{2.8, 2.5, 1.4, 1\} = 2.8$ and $dist_H(G, Q) = \max\{1.4, 1, 2, 1.4\} = 2$. Finally, we obtain $dist_{SH}(Q, G) = \max\{2.8, 2\} = 2.8$.

Then we discuss the extensions of our techniques for similarity search on point groups under the symmetric Hausdorff distance. First, consider the representative-based lower bound technique in

Algorithm 2 Similarity search

```
Require: query point group $Q$, dataset $\mathcal{D}_{groups}$, number of results $K$, parameters $\alpha, \beta, \gamma$
1:  $resultSet \leftarrow \emptyset$
2:  $dist_{kBest} \leftarrow \infty$
3:  $candSet \leftarrow \mathcal{D}_{groups}$
4:  $Q' \leftarrow$ Greedy-Representatives$(Q, \alpha)$        ▷ Section 3.1
5:  $candSet \leftarrow$ Find-Candidates$(\mathcal{D}_{groups}, \beta, \gamma)$        ▷ Section 3.4
6:  sort $candSet$ in ascending order of lower bound values
7:  for each point group $G_{key} \in candSet$ do
8:      if $G_{key}.LB \geq dist_{kBest}$ then
9:          break
10:     $G_{key}.LB \leftarrow$ IncDist$_H(Q', G_{key}, dist_{kBest})$        ▷ Section 3.1
11:     if $G_{key}.LB < dist_{kBest}$ then
12:         if $|G_{key}| \leq 10$ then
13:             $temp \leftarrow$ IncDist$_H(Q - Q', G_{key}, dist_{kBest})$
                                                            ▷ Section 3.2
14:         else
15:             $temp \leftarrow$ Opt-IncDist$_H(Q - Q', G_{key}, dist_{kBest})$
                                                            ▷ Sections 3.2, 3.3
16:         $G_{key}.dist \leftarrow \max\{G_{key}.LB, temp\}$
17:         if $G_{key}.dist < dist_{kBest}$ then
18:             update $dist_{kBest}$ and $resultSet$
19: return $resultSet$
```

Algorithm 3 IncDist$_H$

```
Require: query point group $Q$, point group $G$, threshold $dist_{kBest}$
1:  $d_{max} \leftarrow 0$
2:  for each point $q_i \in Q$ do
3:      $d_j \leftarrow \min_{p_j \in G} dist(q_i, p_j)$
4:      if $d_j \geq d_{max}$ then
5:          $d_{max} \leftarrow d_j$
6:          if $d_{max} \geq dist_{kBest}$ then
7:              return $d_{max}$
8:  return $d_{max}$
```

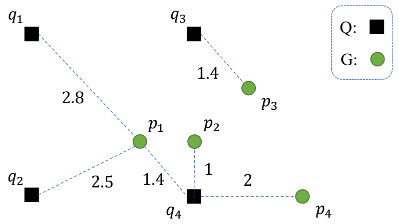


Figure 9: Symmetric Hausdorff distance

Section 3.1. The lower bound function becomes:

$$LB_{rep}(Q, G_{key}) = \max\{dist_H(Q', G_{key}), dist_H(G'_{key}, Q)\} \quad (3)$$

where Q' is a representative subset of Q and G'_{key} is a representative subset of G_{key}. This requires us to obtain representatives for each point group in the dataset $\mathcal{D}_{groups}$. Fortunately, this task could be done in the preprocessing stage because the representatives of data point groups are independent of the query point group.

Second, we adapt the incremental computation technique in Section 3.2. The straightforward way is to first apply incremental computation on $dist_H(Q', G_{key})$ and then apply incremental computation on $dist_H(G'_{key}, Q)$. However, this may prevent us from getting a tight bound early, if the larger value is contributed by the second term. To resolve this issue, our idea is to interleave the incremental computation of $dist_H(Q', G_{key})$ and $dist_H(G'_{key}, Q)$. For example, we run the first iteration in $dist_H(Q', G_{key})$ (by processing the first point in Q'), then run the first iteration in $dist_H(G'_{key}, Q)$, then run the second iteration in $dist_H(Q', G_{key})$, etc.

The candidate retrieval technique in Section 3.4 is directly applicable. To adapt the optimization technique in Section 3.3, we also consider pruning opportunities within the incremental computation of $dist_H(G'_{key}, Q)$. For example, given the current point $p_j \in G'_{key}$, we need to quickly find a point $q_i \in Q$ such that $dist(p_j, q_i)$ is as small as possible.

4 EXPERIMENTAL STUDY

We conducted all the experiments on MacOS platform, with Intel I7-6920HQ processor and 16GB memory. In the experiments, we use the Twitter dataset [9] that contains 40 million tweets collected from April to October in 2012. Each tweet in the dataset [9] consists of five attributes including *date, time, keyword-list, longitude* and *latitude*. In order to obtain keyword-induced point groups, we first remove all non-character keywords (e.g. "??") and transform all remaining ones into lower case. Then, for a tweet represented as < *date, time, keyword-list, longitude, latitude* >, we insert a point *p(longitude, latitude)* into the point groups corresponding to each keyword in *keyword-list*. In this way, we get 1,783,340 point groups from the dataset and each group corresponds to exactly one keyword. The average number of points contained in each point group is 62 and the distribution of group size is shown below:

Table 6: Distribution of the point group size

range of group size	number of point groups	percentage
[1, 9]	1,638,758	91.89%
[10, 99]	109,544	6.14%
[100, 999]	25,942	1.45%
[1000, 9999]	7,368	0.41%
[10000, 99999]	1,579	0.09%
[100000, 999999]	149	0.01%

To generate the query point groups, we extract 50 point groups belonging to five different types (i.e., Q_1, Q_2, Q_3, Q_4, Q_5), and each type consists of 10 point groups. We ensure that, for each query point group of type Q_i, its size is within the range $[10^{i-1}, 10^i)$. We measure the response time of type Q_i as the average response time of each query point group belonging to type Q_i.

Note that in the experimental setting of [2], a simple nested for-loop is used for exact Hausdorff distance calculation. For the sake of fairness, we use a more efficient method proposed in [20] to calculate exact Hausdorff distance in both [2] and our methods. We denote this optimized competitor as GIS2011. The preprocessing time (e.g., the time for building R-tree and generating MBRs in [2]) is not included in the response time.

As shown in Table 7, we consider variants of our proposed techniques to measure their effectiveness. We vary parameters to test the performance of different methods, and Table 8 illustrates the involved parameters and their default values. Those default values are obtained through experimental tests, which will be discussed in Section 4.1.

Table 7: Competitors

method	meaning
GIS2011	the state-of-the-art method
RepLB	representative lower bound only (Sections 3.1, 3.2)
Opt-RepLB	optimized RepLB (Section 3.3)
Cand-RepLB	RepLB + candidate retrieval (Section 3.4)
Comb-RepLB	all proposed techniques (Section 3.5)

Table 8: Experimental parameters

parameter	meaning	default value
K	number of nearest neighbors	10
α	$\frac{\text{number of representatives}}{\text{size of query point group}}$	5%
β	$\frac{\text{number of sample point groups}}{\lvert D_{groups} \rvert}$	10%
γ	number of filtering points	2

4.1 Robustness Experiments

In this section, we examine the robustness of our proposed methods and we tune the values of parameters α, β, γ. In general, we achieve reasonable good performance with a wide range of parameter values. For example, α within 1% to 30%, β within 5% to 50% and γ within 2 to 5. During tuning each parameter, we follow the default settings for other parameters.

Tuning α. Figure 10 depicts the performance of RepLB over different values of α. In the beginning, when α is smaller than 5% , the response time drops dramatically with the increase of α. After α reaches 5%, that is, at least 5% of the query points are selected as representatives, the response time changes slightly as α increases further. The reason is that, our greedy selection method can find effective representatives of the query point group, rendering the lower bounds for data point groups quite tight even in the very beginning. Therefore, we use 5% as the default value of α.

Tuning β. We evaluate both the size of the candidate set and the response time for the effect of β. The size of candidate set drops almost linearly as the value of β increases (Figure 11a), while the response time changes slightly after β reaches 10% (Figure 11b). The reason is that, with the increase of β, the filtering threshold $dist_{kBest}$ decreases, which means the range used to retrieve candidate sets shrinks, leading to the drop of the candidate set size (see Section 3.4). But for the response time, large data point groups cover large regions such that they are difficult to be pruned even with large β, and the response time is dominated by those large point groups. Therefore, the response time changes slightly after β reaches 10%. To this end, we set the default value of β as 10%.

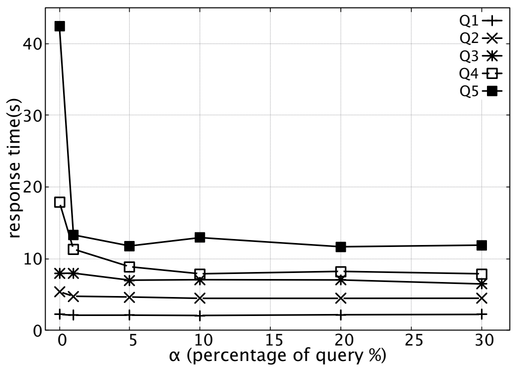
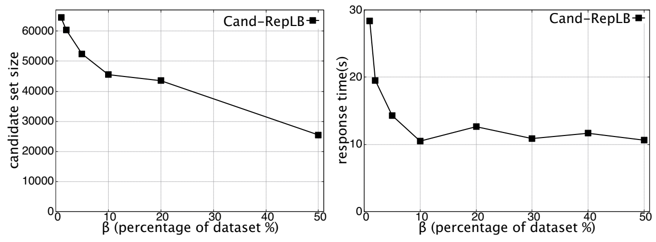


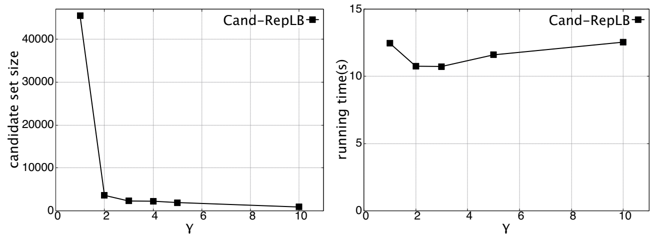
Figure 10: Tuning α



(a) size of candidate set (b) response time

Figure 11: Tuning β

Tuning γ. In the beginning, both the size of candidate set and the response time drop dramatically with the increase of γ (Figure 12). But after γ reaches 2, the remaining candidate point groups are large enough to be included within the search range from most points in the query point group. Therefore, the candidate set size remains stable after γ reaches 2. When γ increases from 1 to 2, the response time decreases as the candidate size shrinks. However, when γ further increases, the overhead in retrieving candidate groups becomes higher. Therefore, we set the default value of γ as 2.



(a) candidate set size (b) response time

Figure 12: Tuning γ

Effect of proposed techniques. Figure 13 depicts the effect of our proposed techniques in terms of response time. RepLB can solve the similarity search queries within less than 3 and 12 seconds for small (query type Q_1) and large (query type Q_5) query point groups, respectively. Opt-RepLB and Cand-RepLB can achieve faster response times with optimization and candidate retrieval techniques. Specifically, the optimization technique brings a maximum 17% improvement compared with RepLB (for query type Q_3) and candidate retrieval technique brings a maximum 40% improvement (for query type Q_1). Finally, Comb-RepLB combines all our proposed techniques and can achieve the best performance. In the following, we use Comb-RepLB as our method to compare with GIS2011. And we set α, β, γ to 5%, 10%, 2 in Comb-RepLB, respectively.

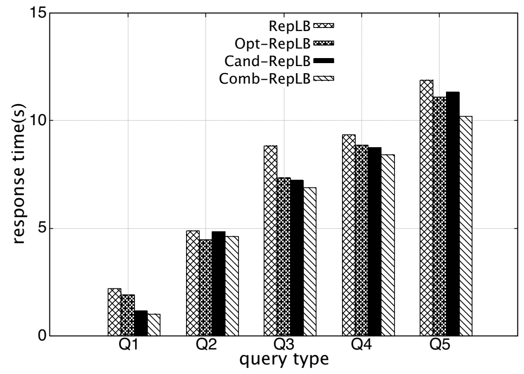


Figure 13: Effect of proposed techniques

4.2 Tuning the state-of-the-art method

The state-of-the-art method GIS2011 [2] requires to specify the amount of MBRs for calculating enhanced lower-bound Hausdorff distances. The number of MBRs affects the tradeoff between tightness of the lower bound and the computation overhead to retrieve the lower bound. For fairness, we tune the number of MBRs for GIS2011 in the experiment.

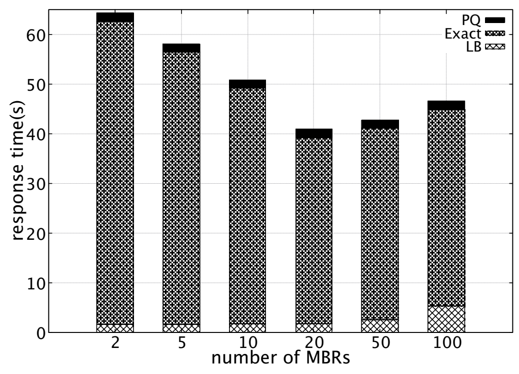


Figure 14: Tuning the setting for GIS2011

As shown in Figure 14, we test the number of MBRs from 2 to 100, and find that 20 MBRs could offer the overall best performance for

GIS2011 method. Thus, we set the number of MBRs in GIS2011 as 20. To verify the correctness of our setting, we check the time spent on different steps of GIS2011, including lower-bound calculation, exact distance calculation and time in priority queue. Figure 15 plots the distribution of time for each step, which is consistent to the result in [2].

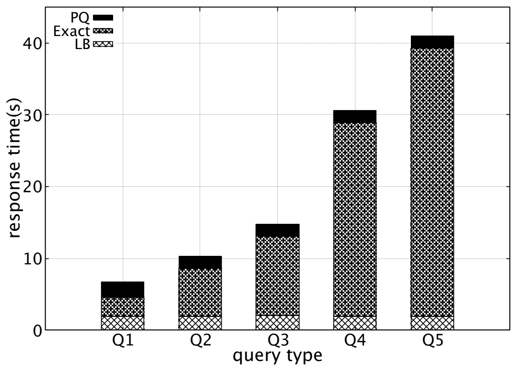


Figure 15: Verifying the setting for GIS2011

4.3 Performance comparison

In this section, we compare our proposed method Comb-RepLB and GIS2011 under several different settings.

Varying query size. Figure 16 shows that, in terms of response time, our proposed Comb-RepLB outperforms GIS2011 in all kinds of query sizes. Specifically, Comb-RepLB is 2.3X to 6.6X faster than GIS2011. Such speed-up mainly attributes to the tight representative-based lower bound, which is especially effective in our scenario of heavy overlapping point groups. In addition, even if the point group cannot be pruned using our lower bound, unlike the lower-bound techniques in GIS2011 [2], the calculation of our lower bound will not be wasted, as it can be recycled to complete the remaining calculation for the exact Hausdorff distance. Moreover, our proposed optimization technique and candidate retrieval technique improve the performance of Comb-RepLB further.

Varying K. Figure 17 depicts the effect of K in terms of response time, and Comb-RepLB outperforms GIS2011 in all test values K. When $K = 1$, both methods are very fast because of the small threshold distance of 1–NN. When K increases to 2, the response time of GIS2011 increases dramatically, which is caused by the loose MBR-based lower bound . When K becomes larger than 2, the response times grow almost linearly to K for both Comb-RepLB and GIS2011.

Varying $|\mathcal{D}_{groups}|$. Figure 18 illustrates the performances of Comb-RepLB and GIS2011 under different dataset sizes. Comb-RepLB outperforms GIS2011 in all test sizes $|\mathcal{D}_{groups}|$. With the increase of $|\mathcal{D}_{groups}|$, the response time of GIS2011 grows much faster than Comb-RepLB. This is because our proposed representative lower-bound, optimization technique and candidate retrieval technique work effectively.

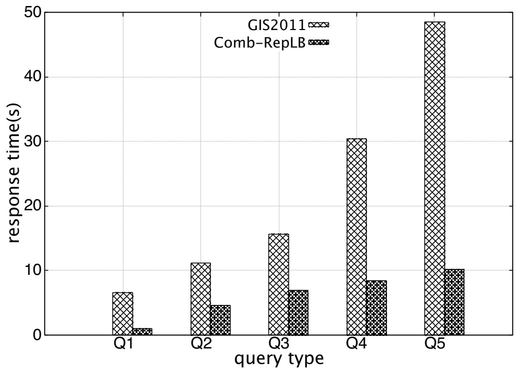


Figure 16: Performance comparison with varying query size

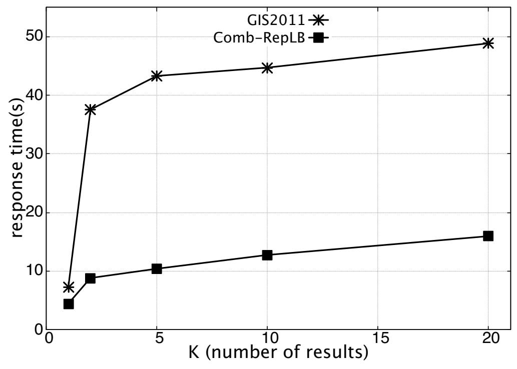


Figure 17: Performance comparison with varying K

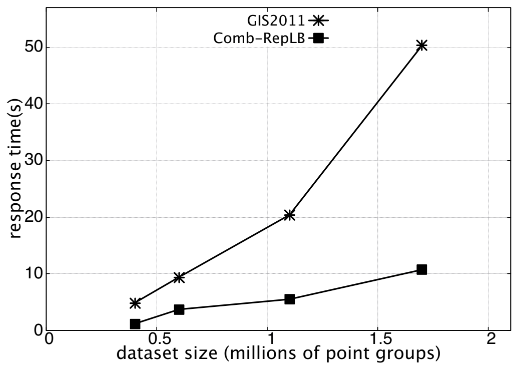


Figure 18: Performance comparison with varying $|\mathcal{D}_{groups}|$

5 RELATED WORK

In this paper, we adopt the Hausdorff distance to measure the dissimilarity between two point groups. The Hausdorff distance has been extensively used in various application domains, such as image recognition, model rendering, spatial querying [3, 14, 18, 21].

Different methods have been proposed to speed up exact Hausdorff distance calculation [4, 10, 18–20], for example, by using R-trees [18] or imposing random ordering within a point group [20]. These works are orthogonal to our problem. We focus on developing lower bound functions and filtering techniques to prune unpromising point groups in similarity search.

Similarity search on point groups has been studied in [2], which is the closest work to our problem. The state-of-the-art solution [2] employs R-trees for indexing data point groups as well as the query point group. Then, the solution exploits MBRs in R-trees to compute lower-bound Hausdorff distances for ordering candidate point groups and pruning unpromising ones. However, unlike [2], our point groups are defined based on geo-textual data (e.g., Twitter data) and the locations within the same point group are required to share the same keyword. As discussed in the introduction, this setting causes many point groups to span over wide areas, thus degrading the effectiveness of the MBR-based lower bounds in [2]. As opposed to using MBRs, our solution utilizes a subset of the query point group to compute lower bound distances.

Processing spatial-keyword queries is a popular research area in the last two decades. Their approaches can be classified as *spatial-first* approaches and *keyword-first* approaches [17]. Spatial-first approaches apply spatial predicate on spatial indexes (e.g., R-tree [6]) to obtain candidates, and then filter candidates by the keyword predicate. Furthermore, spatial indexes can be augmented with keyword information to improve the effectiveness of pruning [11, 15]. Keyword-first approaches apply inverted file [23] or bitmap [12] to retrieve candidates by the keyword predicate, and then refine them by the spatial predicate. Nevertheless, none of the above methods consider similarity search on point groups.

Many applications on Twitter data have been studied in the literature. Real-time discovery of local events is useful in crime monitoring, disaster alarming, and activity recommendation. Krumm and Horvitz [16] conduct time series analysis over geo-tagged tweet volumes to detect local events accurately. To improve accuracy and efficiency further, Zhang et al. [22] design a novel authority measure that captures geo-topic correlations among tweets, and develop a pivot-based searching algorithm to identify local events. Bursty words play an important role in social media as it might reveal important trending topics that may trigger actions. Abdelhaq et al. [1] propose a sliding window approach to detect bursty words over continuous streams of Twitter texts. Large volume of tweets can also be used in recommending topics [8], ranking financial tweets [7], and classifying a user's political orientation [13]. The above problems are different from our problem, i.e., similarity search on keyword-induced point groups.

6 CONCLUSIONS

We propose the concept of keyword-induced point groups and study the similarity search problem on such point groups under the Hausdorff distance measure. Although there exists a solution for our problem, it has not considered the characteristics of keyword-induced point groups and suffers from overlapping point groups. The novelty of our solution is that it does not rely on minimum bounding rectangles. Experimental results on Twitter data demonstrate that our solution is faster than the state-of-the-art by up to 6 times.

In future, we will study the similarity join problem on keyword-induced point groups, e.g., finding all pairs of keywords such that their corresponding point groups have Hausdorff distance below a given distance threshold.

ACKNOWLEDGMENTS

This work was supported by GRF 152196/16E from the Hong Kong RGC.

REFERENCES

[1] Hamed Abdelhaq, Michael Gertz, and Christian Sengstock. 2013. Spatio-temporal Characteristics of Bursty Words in Twitter Streams. In *ACM SIGSPATIAL*.
[2] Marco D. Adelfio, Sarana Nutanong, and Hanan Samet. 2011. Similarity search on a large collection of point sets. In *ACM SIGSPATIAL*. 132–141.
[3] Helmut Alt, Bernd Behrends, and Johannes Blömer. 1995. Approximate Matching of Polygonal Shapes. *Ann. Math. Artif. Intell.* 13, 3-4 (1995), 251–265.
[4] Mikhail J. Atallah. 1983. A Linear Time Algorithm for the Hausdorff Distance Between Convex Polygons. *Inf. Process. Lett.* 17, 4 (1983), 207–209.
[5] Jie Bao, Yu Zheng, David Wilkie, and Mohamed F. Mokbel. 2015. Recommendations in location-based social networks: a survey. *GeoInformatica* 19, 3 (2015), 525–565.
[6] Ariel Cary, Ouri Wolfson, and Naphtali Rishe. 2010. Efficient and scalable method for processing top-k spatial boolean queries. In *SSDBM*.
[7] Diego Ceccarelli, Francesco Nidito, and Miles Osborne. 2016. Ranking Financial Tweets. In *SIGIR*.
[8] Chen Chen, Hongzhi Yin, Junjie Yao, and Bin Cui. 2013. TeRec: A Temporal Recommender System over Tweet Stream. *Proc. VLDB Endow.* 6, 12 (2013), 1254–1257.
[9] Lisi Chen, Gao Cong, Xin Cao, and Kian-Lee Tan. 2015. Temporal spatial-keyword top-k publish/subscribe. In *ICDE*. 255–266.
[10] Krzysztof Chris Ciesielski, Xinjian Chen, Jayaram K. Udupa, and George J. Grevera. 2011. Linear Time Algorithms for Exact Distance Transform. *Journal of Mathematical Imaging and Vision* 39, 3 (2011), 193–209.
[11] Gao Cong, Christian S Jensen, and Dingming Wu. 2009. Efficient retrieval of the top-k most relevant spatial web objects. *Proceedings of the VLDB Endowment* 2, 1 (2009), 337–348.
[12] Ian De Felipe, Vagelis Hristidis, and Naphtali Rishe. 2008. Keyword search on spatial databases. In *ICDE*.
[13] Anjie Fang, Iadh Ounis, Philip Habel, Craig Macdonald, and Nut Limsopatham. 2015. Topic-centric Classification of Twitter User's Political Orientation. In *SIGIR*.
[14] Michael Guthe, Pavel Borodin, and Reinhard Klein. 2005. Fast and Accurate Hausdorff Distance Calculation between Meshes. *Journal of WSCG* 13, 2 (2005), 41–48.
[15] Ramaswamy Hariharan, Bijit Hore, Chen Li, and Sharad Mehrotra. 2007. Processing spatial-keyword (SK) queries in geographic information retrieval (GIR) systems. In *SSDBM*.
[16] John Krumm and Eric Horvitz. 2015. Eyewitness: Identifying Local Events via Space-time Signals in Twitter Feeds. In *ACM SIGSPATIAL*.
[17] Taesung Lee, Jin-woo Park, Sanghoon Lee, Seung-Won Hwang, Sameh Elnikety, and Yuxiong He. 2015. Processing and Optimizing Main Memory Spatial-keyword Queries. *Proc. VLDB Endow.* 9, 3 (2015), 132–143.
[18] Sarana Nutanong, Edwin H. Jacox, and Hanan Samet. 2011. An Incremental Hausdorff Distance Calculation Algorithm. *PVLDB* 4, 8 (2011), 506–517.
[19] Dimitris Papadias, Yufei Tao, Kyriakos Mouratidis, and Chun Kit Hui. 2005. Aggregate nearest neighbor queries in spatial databases. *ACM Trans. Database Syst.* 30, 2 (2005), 529–576.
[20] Abdel Aziz Taha and Allan Hanbury. 2015. An Efficient Algorithm for Calculating the Exact Hausdorff Distance. *IEEE Trans. Pattern Anal. Mach. Intell.* 37 (2015), 2153–2163.
[21] Min Tang, Minkyoung Lee, and Young J. Kim. 2009. Interactive Hausdorff distance computation for general polygonal models. *ACM Trans. Graph.* 28, 3 (2009), 74:1–74:9.
[22] Chao Zhang, Guangyu Zhou, Quan Yuan, Honglei Zhuang, Yu Zheng, Lance Kaplan, Shaowen Wang, and Jiawei Han. 2016. GeoBurst: Real-Time Local Event Detection in Geo-Tagged Tweet Streams. In *SIGIR*.
[23] Yinghua Zhou, Xing Xie, Chuang Wang, Yuchang Gong, and Wei-Ying Ma. 2005. Hybrid index structures for location-based web search. In *CIKM*.