Hybrid-LSH for Spatio-Textual Similarity Queries

Mingdong Zhu[1], Derong Shen[1], Ling Liu[2], and Ge Yu[1]

[1] Northeastern University, China,
mingdongzhu@hotmail.com, {shenderong, yuge}@ise.neu.edu.cn
[2] Georgia Institute of Technology, USA,
lingliu@cc.gatech.edu

Abstract. Locality Sensitive Hashing (LSH) is a popular method for high dimensional indexing and search over large datasets. However, little efforts have put forward to utilizing LSH in mobile applications for processing spatio-textual similarity queries, such as find nearby shopping centers that have a top ranked hair salon. In this paper, we present hybrid-LSH, a new LSH method for indexing data objects according to both their spatial location and their keyword similarity. Our hybrid-LSH approach has two salient features: First our hybrid-LSH carefully combines the spatial location based LSH and textual similarity based LSH to ensure the correctness of the spatial and textual similarity based NN queries. Second, we present an adaptive query-processing model to address the fixed range problem of traditional LSH and to handle queries with varying ranges effectively. Extensive experiments conducted on both synthetic and real datasets validate the efficiency of our hybrid LSH method.

Keywords: similarity query, hybrid LSH, spatio-textual query

1 Introduction

Location-based services have become more and more prevalent and have attracted significant attentions from both industry and academic community. Apple has an application to locate frequently used software; Yelp finds nearby restaurants of interest; and Facebook and FourSquare offers its members the capability to find his or her nearby (local) friends or points of interest.

One obvious way [1] to combine spatial distance and textual similarity is to use a spatial index to a spatial partition of the large dataset, which is most relevant to the spatial location of the query issuer, and for the given partition a string index is used to filter out those irrelevant objects by keywords matching and then rank the results using a hybrid ranking function that can combine spatial distance and textual similarity [2–4]. Spatio-textual data ordinarily are high-dimensional. For example, a typical microblog with location has 2 coordinates and some keywords, say 100, then the whole microblog, simultaneously considering location and text, contains $2\times100=200$

features(dimensions). Due to the curse of dimensionality, many traditional methods are not efficient. It's known that locality sensitive hashing(LSH) is a good method for similarity queries on high-dimensional data. However, none of previous work, to the best of our knowledge, has explored the feasibility of utilizing LSH for processing spatio-textual similarity queries. We argue that LSH is an attractive alternative method for processing spatio-textual similarity queries: First, LSH-based methods have no hierarchical structure, thus are easy to be maintained and scaled. Second, LSH can be directly used to hash spatially and textually similar objects to the same buckets, which can be obtained with less I/O cost.

In this paper, we design hybrid-LSH to process spatio-textual similarity queries, and the method treats spatial information and textual content of an object as a whole, rather than builds indices separately and combines two sets of query results when there is a query. The first challenge is how to design the hybrid-LSH such that hash buckets are conducted by considering spatial and textual similarity simultaneously. For each object, one hash value that reflects both spatial information and textual content considered as a whole, should be generated. The second challenge is how to make the hybrid LSH adaptive to spatio-textual similarity queries with different similarity ranges efficiently, because for LSH based methods their sensitive radii are fixed and it's difficult to answer queries with varying ranges.

To address these challenges, we propose a hybrid-LSH structure which considers both spatial and textual similarity, so that it is with high probability that spatially and textually similar objects are stored in the same bucket and can be found with one disk I/O. Then we present adaptive algorithms for queries with varying ranges. In addition, the hybrid-LSH's effectiveness and algorithms' accuracy are guaranteed by theoretical analysis. To summarize, we make the following contributions.

- By simultaneously considering spatial and textual similarity, we propose a hybrid-LSH and prove its theory features.
- To process query with varying ranges on the hybrid-LSH, we provide adaptive algorithms to process the query.
- We conduct extensive experiments on real and synthetic datasets in a distributed environment. Experimental results confirm the scalability and effectiveness of our approach.

2 OVERVIEW

Reference Model. Both the object model and the query model are defined with spatial location information and textual content consisting of keyword tokens. We assume that the spatial information and the textual content of objects are independent.

Formally, let P denote the universe of discourse, namely the set of spatial objects. Each object $p \in P$ is defined as a two-element tuple (loc, tok), where *p.loc* is the spatial location information of object p and *p.tok* is a set of tokens which represent the textual description of p. In order to compute spatio-textual similarity between two objects, say p_1 and p_2, we define a spatio-textual distance metric that combines spatial and textual similary through a weight parameter α, as shown in Eqn. 1.

$$\mathrm{DistST}(p_1, p_2) = \alpha * \mathrm{DistS}(p_1, p_2) + (1 - \alpha) * \mathrm{DistT}(p_1, p_2) \tag{1}$$

$$\mathrm{DistS}(p_1, p_2) = \frac{\mathrm{dist}(p_1.loc, p_2.loc)}{dmax - dmin} \tag{2}$$

$$\mathrm{DistT}(p_1, p_2) = 1 - \frac{(p_1.tok \cap p_2.tok)}{(p_1.tok \cup p_2.tok)} \tag{3}$$

We use the normalized Euclidean distance of objects $p_1, p_2 \in P$, denoted as $\mathrm{dist}(p_1.loc, p_2.loc)$, to compute the spatial distance DistS, as shown in Eqn.2. *dmax* and *dmin* in Eqn.2 denote the maximum and minimum distance for pairs of objects in P. We use Jaccard distance [5] to measure the distance of textual similarity as shown in Eqn.3. Note that our hybrid LSH method is generic and independent of the specific distance functions used and thus can incorporate other spatial distance function and textual similarity distance function. For simplicity, $\mathrm{B}(q, D)$ denotes object set $\{o \in P | DistS(o.loc, q.loc) \leq D\}$, similarly, $\mathrm{B}(q, R)$=$\{o \in P | DistT(o.tok, q.tok) \leq R\}$, and $\mathrm{B}(q, D, R)$= $\{o \in P |\ \mathrm{DistS}(o, q) \leq D$ and $\mathrm{DistT}\ (o, q) \leq R\ \}$.

Given that spatial location based similarity is defined based on Euclidean distance and textual similarity is defined based on Jaccard distance, [6] and [7] describe the construction of an LSH family for Euclidean distance and the construction of an LSH family for Jaccard distance respectively.

Because the traditional (R, c)-NN problem [6,8] just adapts to objects with single data type. we extend the (R, c)-NN problem to (D, R, c)-NN for spatio-textual objects, which return an object $o \in \mathrm{B}(q, cD, cR)$ if there exists an object $o^* \in \mathrm{B}(q, D, R)$.

Related Work. There are many studies on spatial textual similarity query processing [2,3,9]. A good survey of techniques can be found in [10]. Generally they can be classified into two categories: tree-like style and grid style. Specifically for tree-like style, [2] proposes a new hybrid index structure Inverted File Quad-tree (IQ-tree) to manage a stream of Boolean range continuous(BRC) queries on a stream of incoming. [11] proposes a new indexing framework for processing the location-aware text retrieval query. [3] proposes a hybrid indexing structures called Intersection-Union-R tree (IUR-tree) and an efficient approach that take into account the fusion of location proximity and document similarity. For grid style, this category of indices combines a grid index with a text index (e.g., the inverted file). For example, [4] proposes a spatio-textual similarity search method (SEAL), which is a filter-and-verification framework.

3 Hybrid-LSH

In this section, we introduce the construction of hybrid-LSH. Concretely, each object $o \in P$ consists of spatial information s and textual content t. Assume that the following three parameters are provided: the spatial range d, the textual similarity range r, and the approximation factor $c > 1$. The hybrid LSH construction algorithm works as follows: For s we can find (d, cd, sp_1, sp_2)-sensitive LSH family, denoted by sH, for Euclidean distance (recall Section 2), while for t we can obtain (r, cr, tp_1, tp_2)-sensitive LSH family, denoted by tH, for Jaccard distance (recall Section 2). We combine the two hash families. In particular, let k_1 and k_2 denote the number of hash functions generated in sH and tH respectively. We define an LSH function family G= g: $S \to U^{(k_1+k_2)}$ with $k_1 + k_2$ hash functions such that g(o)={sh$_1$(o),$\cdots$,sh$_{k_1}$(o), th$_1$(o), $\cdots$, th$_{k_2}$(o)}, where th$_i$ $\in$tH and sh$_i$ $\in$sH. Let L be an integer, we choose L functions g$_1$,$\cdots$, g$_L$ from G independently and uniformly at random. During preprocessing, we store an indicator of object o in the bucket g$_i$(o) for i=1,$\cdots$, L. We define the spatial component as g$_s$(o), i.e., g$_s$(o)={sh$_1$(o), $\cdots$,sh$_{k_1}$(o)} and the textual component as g$_t$(o), i.e., g$_t$(o)={th$_1$(o), $\cdots$, th$_{k_2}$(o)}. Fig.1 is an illustration of hybrid-LSH.

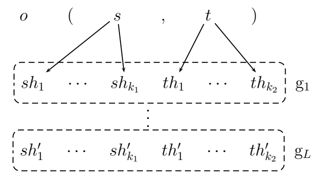

o (s , t)
sh_1 $\cdots$ sh_{k_1} th_1 $\cdots$ th_{k_2}
g$_1$
sh'_1 $\cdots$ sh'_{k_1} th'_1 $\cdots$ th'_{k_2}
g$_L$

Fig. 1. An illustration of hybrid-LSH

Intuitively, hybrid-LSH is composed by k_1 hash values from (d, cd, sp_1, sp_2)-sensitive LSH and k_2 hash values from (r, cr, tp_1, tp_2)-sensitive LSH. Thus, hybrid-LSH is defined as $(d, r, c,\ \{sp_1, tp_1\},\ \{sp_2, tp_2\})$-sensitive hybrid-LSH. Because sp_1, sp_2, tp_1 and tp_2 are determined by d, r and c, $(d, r, c,\ \{sp_1, tp_1\},$ $\{sp_2, tp_2\})$-sensitive is simplified as (d, r, c)-sensitive hybrid-LSH when no confusion occurs.

Base on hybrid-LSH, in order to process a (D, R, c)-NN query with query object q, simplified as query q, we first generate L hash value of q, i.e., g$_1$(q), $\cdots$, g$_L$(q), then search corresponding buckets of the hash values and randomly check $C * L$ objects in the buckets where C is a constant number. Let $o_1, \cdots, o_{C*L}$ be the checked objects. For any o_i, if DistS$(o_i, q) < cD$ and DistT$(o_i, q) < cR$, then we return YES and o_i, otherwise we return NO. Because there are probably lots of objects in all the buckets, it saves lots of computation to only check constant number of objects.

To ensure the correctness of the algorithm, the parameters k_1, k_2 and L are chosen so as to ensure the following properties hold with constant probability:

P_1 If there exists object o such that $\mathrm{DistS}(o,q) < D$ and $\mathrm{DistT}\ (o,q) < R$, denoted as $\mathrm{B}(q,D,R)$, then $\mathrm{g}_i(o)$=$\mathrm{g}_i(q)$ for some $i = 1,\cdots,L$.

P_2 The total number of collisions of q with the number of objects, which do not belong to $\mathrm{B}(q,cD,cR)$, is less than $3L$.

P_1 ensures objects who satisfy the query at least collide once for the L hash values. P_2 ensures if there is object who satisfies the query algorithm can find the object after checking $3L$ objects.

Theorem 1. *setting* $k_1 = \log_{sp_2}(1-\sqrt{1-\frac{1}{n}})$, $k_2 = \log_{tp_2}(1-\sqrt{1-\frac{1}{n}})$ *and* $L = (1-\sqrt{1-\frac{1}{n}})^{-(\log_{sp_2} sp_1 + \log_{tp_2} tp_1)}$ *guarantees that properties* P_1 *and* P_2 *hold with constant probability.*

Proof. Let P_1 hold with probability p_1 and P_2 hold with p_2. Without loss of generality, assume that there is an object o^* which satisfies $\mathrm{DistS}(o^*.loc, q.loc) < D$ and $\mathrm{DistT}(o^*.tok, q.tok) < R$. Set $k_1 = \log_{sp_2}(1-\sqrt{1-\frac{1}{n}})$ and $k_2 = \log_{tp_2}(1-\sqrt{1-\frac{1}{n}})$. Use P(A) to denote the probability of event A. $\mathrm{P}(\mathrm{g}(o')$=$\mathrm{g}(q)$ & $o' \in (P-\ \mathrm{B}(q,cD)\cap\mathrm{B}(q,cR))$ (denoted as Pa) is not larger than P $(\mathrm{g}_s(o')$=$\mathrm{g}_s(q)$ & $o' \in (P-\mathrm{B}(q,cD)) \mid \mathrm{g}_t(o') = g_t(q)$ & $o' \in (P-\mathrm{B}(q,cR))$ (denoted as Pb). $Pb = 1-(1-sp_2^{k_1})(1-tp_2^{k_2}) = \frac{1}{n}$. so $Pa < \frac{1}{n}$, and then the expected number of objects allocated for g_i which don't satisfy the query condition is less than 1.5. The expected number of the objects for all g_i doesn't exceed $1.5L$, According to the Markov inequality, the probability that this number exceed $3L$ is less than $\frac{1}{2}$. so P_2 follows. The probability of $\mathrm{g}(o^*)$=g(q) is ${sp_1}^{k_1} * {tp_1}^{k_2}$=$(1-\sqrt{1-\frac{1}{n}})^{\log_{sp_2} sp_1 + \log_{tp_2} tp_1}$. By setting $L = (1-\sqrt{1-\frac{1}{n}})^{-(\log_{tp_2} tp_1 + \log_{sp_2} sp_1)}$, the probability that all $\mathrm{g}_i(o^*) \neq \mathrm{g}_i(q)$ is less than $\frac{1}{e}$. so the P_1 holds with the probability 1-$\frac{1}{e} > \frac{1}{2}$. □

4 Adaptive (D,R,c)-NN Query Processing with Hybrid-LSH

This section focuses on describing how to split a (D,R,c)-NN query into multiple subqueries with smaller query ranges such that each subquery can be processed directly using our hybrid LSH index. Recall Section 3, we show the structure of (d,r,c)-sensitive hybrid-LSH. For (D,R,c)-NN queries where $D \geq d$ and $R \geq r$, we need to decompose each (D,R,c)-NN query into multiple subqueries with spatial range d and textual range r.

4.1 Adaptive (D,R,c)-NN Query Processing

Let $d < D$ and $r < R$, and we build a hybrid-LSH with sufficiently small d and r so that ranges from query are bigger than them. The intent of the adaptive (D,R,c)-NN query method is to transform an original query into several queries with small query range which can be processed by the constructed hybrid-LSH. Then we give a formal definition of the transform as follows.

Definition 1. (*LSH query transform*) *LSH query transform is based on a single type (r_1, r_2, p_1, p_2)-sensitive LSH. If a query (R, c)-NN can be transformed into a query set S which contains several queries based on (r_1, r_2, p_1, p_2)-sensitive LSH, where $r_1 < R$, $r_2 < cR$, and S satisfies the following two properties, then S is a (r_1, r_2, p_1, p_2)-sensitive LSH query transform of q.*

$P1$ *the area or content which is covered by R in q is contained by the area or content which is covered by r_1 in S.*
$P2$ *the area or content which is covered by r_2 in S is contained by the area or content which is covered by cR in q.*

Then we show the effect of LSH query transform in Lemma 1, compared with the original query.

Lemma 1. *If there is an answer for a (R, c)-NN query, then a (r_1, r_2, p_1, p_2)-sensitive LSH query transform of the (R, c)-NN query can return a c-approximate answer with constant probability.*

Proof. Set QT be a LSH query transformation of a (R, c)-NN query q, where $QT = \{qt_i | 0 < i < |QT|\}$. If there is an object $o^* \in$B(q, R), according to P1 in definition 1, then o^* locate in the query range of at least one query in QT, say qt_i. In addition, the objects which are outside of B(q, cR), denoted as O, are outside of B(qt_i, r). According to property 2 of LSH (R, c)-NN query in the [8], the number of collisions of st_i and O is less than $C * L$, So after checking $C * L$ objects st_i can return a c-approximate answer with constant probability. Then Lemma 1 follows. □

According to Theorem 1 we can use a LSH query transform to processing queries with varying ranges. However it is nontrivial to construct LSH query transform for spatial information or textual content. Now we present the LSH query transform methods for spatial information and textual content respectively.

For spatial information, it is similar with a disk covering problem to find a LSH query transform. To the best of our knowledge, optimal solutions of disk covering problems are only available for limited situations. We try to give a general algorithm and show it is a 3-approximate optimal method. Given a (d, cd, p_1, p_2)-sensitive LSH and a (D, c)-NN query with centre coordinate $p = (p_x, p_y)$, where $d < D$, we use several squares with sides $\sqrt{2}d$ to cover the area of B(p, D), as shown in figure 2. The number of squares is $\lceil \frac{2D^2}{d^2} \rceil$. Then the (D, c)-NN queries from central points of the squares is a LSH query transformation, denoted as $ST = \{st_i\}$. Note that because $d < D$, ST is easy to satisfy the condition that the coverage of all queries in ST with radii cd is contained by the circle with radius cD of query q. According to the area formula, the low bound of optimal method is $\lceil \frac{D^2}{d^2} \rceil$. So it is easy to get an corollary in the following.

Corollary 1. *There is a constant c which makes ST a LSH query transformation of query q, and the LSH query transformation is a 3-approximate optimal method.*

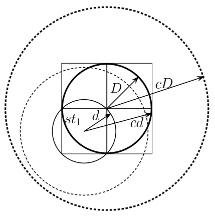


Fig. 2. LSH query transform for spatial data

For textual content, given a (r, cr, p_1, p_2)-sensitive min-hash and a textual (R, c)-NN query q, where $R > r$ and there are TL tokens in q, in order to find similar token sets by (r, cr, p_1, p_2)-sensitive LSH, we should consider two kinds of objects: objects in which the number of tokens are smaller than q and objects in which the number of tokens are bigger than q. For smaller objects, we should get rid of some tokens in q to match them. For an integer m specified later, we generate a query set which consists of all possible combinations of $(TL - m)$ tokens from tokens of q, which is denoted as DS. And for bigger objects, we should add some wildcard tokens in q to mach them. For an integer w specified later, we add w wildcard tokens to q. When q is hashed to a hash value, a wildcard token is hashed to all possible hash values. In this way we get another query set AS. Then by combining DS and AS we get a query set TT.

Let IN and UN be the size of intersection and union of two token sets, respectively. When parameters m and w satisfy the formulae 4, 5 and 6, it ensures that all similar objects of q are covered by the similar objects of query set TT, corresponding to P1 in definition 1. And when parameters m and w satisfy the formula 7, 8 and 9, it ensures that all c approximate similar objects of query set TT are covered by c approximate similar objects of q, corresponding to P2 in definition 1. By selecting the smallest m and w which satisfy the formula, corollary 2 follows.

$$\frac{IN}{UN} \geq 1 - R. \tag{4}$$

$$\frac{IN}{UN - m} \geq 1 - r \tag{5}$$

$$\frac{IN + w}{UN + w} \geq 1 - r \tag{6}$$

$$\frac{IN}{UN} \leq 1 - cR \tag{7}$$

$$\frac{IN}{UN - m} \leq 1 - cr \tag{8}$$

$$\frac{IN + w}{UN + w} \leq 1 - cr \tag{9}$$

Corollary 2. *when $m = \lceil \frac{R-r}{(1-R)(1-r)} TL \rceil$ and $w = \lceil \frac{R-r}{(1-R)r} TL \rceil$, where TL is number of tokens in the query q, query set TT is a LSH query transformation of query q.*

Proof. Due to the limited space, the proof is omitted. □

Based on LSH query transform, the adaptive (D, R, c)-NN query algorithm is straightforward. The idea of the algorithm is to decompose a query for hybrid type data to queries for single type data, then generate LSH query transforms of the queries, join the LSH query transforms of two types, and lastly process the joined queries on hybrid-LSH. Specifically, for a (D, R, c)-NN query, it can be seen as a join of two single type queries (D, c)-NN and (R, c)-NN. First We find a (d, cd, sp_1, sp_2)-sensitive LSH query transform DT of (D, c)-NN and a (r, cr, tp_1, tp_2)-sensitive LSH query transform RT of (R, c)-NN query. By combining the set DT and RT in Cartesian product way, we generate a (d, r, c)-sensitive hybrid-LSH query transform, denoted as DRT. At last we process the DRT in the hybrid-LSH and return the query result.

Theorem 2. *If there is an answer for a (D, R, c)-NN query, the adaptive (D, R, c)-NN query algorithm can return a c-approximate answer with constant probability.*

Proof. The proof is straightfoward. □

4.2 Multiple Adaptive Hybrid-LSHs

Shown in Theorem 1, each query in LSH query transformation at most checks $3L$ objects, so the number of buckets accessed by each query is directly proportional to the number of queries. Hence, the number of queries in hybrid-LSH query transformation is a direct indicator of query cost. The number of queries in hybrid-LSH query transformation is:

$$QN \leq (\frac{2D^2}{d^2} + 1)(C_{TL}^m + H^w) \tag{10}$$

According to Formula 10, when the query range is big, the query cost is very high, so we propose the multiple adaptive hybrid-LSHs.

The intent of the multiple adaptive hybrid-LSHs is to build many adaptive hybrid-LSHs to make the distance between queries and the closet adaptive hybrid-LSH small, which can significantly reduce the cost of query.

Shown in Formula 10, QN is in direct proportion to $\frac{D}{d}$ (spatially) and $(R-r)$ (textually). For (D, R, c)-NN queries where $MinD \leq D \leq MaxD$ and $MinR \leq R \leq MaxR$, we build $\{d_i, r_j, c\}$-sensitive adaptive hybrid-LSHs, where i, j are integers, $i \in (0, \log_b(\frac{MaxD}{MinD}))$, $j \in (0, \frac{MaxR-MinR}{t})$, $d_1 = MinD$, $\frac{d_{j+1}}{d_j} = b$, $r_1 = MinR$, $r_{i+1} - r_i = t$, and b, t are the common ratio and difference respectively.

Corollary 3. *Multiple adaptive hybrid-LSHs uses* $\log_b(\frac{MaxD}{MinD}) * \frac{MaxR-MinR}{t}$ *adaptive hybrid LSHs to process any* (D, R, c)*-NN query, where* $MinD < D < MaxD$ *and* $MinR < R <= MaxR$*, by hybrid-LSH query transformation with at most* $(2b^2+1)(C^m_{TL}+H^w)$*, where* $m = \frac{t}{(1-MaxR)^2}TL$*,* $w = \frac{t}{(1-MinR)(MinR-t)}TL$*.*

Proof. Set multiple adaptive hybrid-LSH MAH=$\{mah_{ij} \mid mah_{ij}$ is (d_i, r_j, c)-sensitive adaptive hybrid-LSH, $0 < i < \log_b \frac{MaxR}{MinR}$, $0 < j < \frac{MaxD-MinD}{t}$ $\}$. There is a query (qd, qr, c)-NN and $mah_{k,s}$ where $k = argmin_{(0<i<\log_b \frac{MaxR}{MinR}, qd>d_i)} \frac{qd}{d_i}$ and $s = argmin_{(0<i<\frac{MaxD-MinD}{t}, qr>r_j)}(qr - r_j)$. For the spatial part, $\frac{qd}{d_k} <= b$, then $(2\frac{D^2}{d^2} + 1) \leq 2b^2 + 1$. For the textual part, $qr - r_s \leq t$ and $qr > r_s$, so $\frac{qr-r_s}{(1-qr)(1-r_s)} < \frac{t}{(1-MaxR)^2}$, which is monotonically increasing function for qr, and $\frac{qr-r_s}{(1-qr)r_s} < \frac{t}{(1-MinR)(MinR-t)}$, which is monotonically decreasing function for qr. So $QN \leq (2b^2 + 1) * (C^m_{TL} + H^w)$ where $m = \frac{t}{(1-MaxR)^2}TL$, $w = \frac{t}{(1-MinR)(MinR-t)}TL$. The Corollary follows. □

5 Experiments

Setup. In order to show scalability and maintainability, we built the adaptive hybrid-LSH (HLSH), multiple adaptive hybrid-LSH(MHLSH), and implemented the proposed (D, R, c)-NN in a distributed setting. For comparison, we also implemented state-of-the-art method SEAL [4] and LSHDSS [12]. SEAL uses hash based hybrid signature to process query, belongs to filter-and-verification framework and is an exact method, so for (D, R, c)-NN queries we stop the algorithm when one object which satisfies query condition is obtained. LSHDSS is a LSH based distributed similarity search algorithm, however it is designed for single data type, so we extended it to support Spatial-textual similarity query by executing separately and combining intermediate result. In this paper we mainly focus on high-dimensional spatio-textual data, so we compared with methods with little global information which can be easy maintained in a distributed setting when there are lots of updates, and due to expensive maintenance cost, methods with tree structures or hierarchical structures were not considered in our experiments.

Two datasets are used, The first one is a real dataset, which contains 0.5 million micro-blogs with location information collected from Sina microblog website, denoted as MicroBlog. According to the rule of microblog, each message must contain less than 140 characters and for MicroBlog, the longest meaningful spatial distance is 30km. After deleting the stop words, the average number of words of a message is 24, and in our experiments each word is taken as a token. The other dataset is a synthetic dataset, denoted as SynSet. SynSet has 1 million objects and objects' tokens are chosen from a word set, and each object's location is generated from a square area of which the side length is 100km.

All experiments are implemented in Cassandra v1.2.6. The Cassandra cluster consists of 10 machines with the same configuration: Intel(R) Core(TM) i7 Quad 870 @2.93GHz CPU and 8 GB RAM, and the same operating system: Ubuntu

10.04. To evaluate the performance of algorithms, we vary one parameter while keeping the others fixed at their default values. We run 20 times for each test, the average result of the query set is reported in the experiments.

Two performance measures are used: number of messages and accuracy. We count the number of messages from sending one query to obtaining result. The number of messages is an important indication of algorithm efficiency. Query accuracy [6] is used to measure the quality of the results. Specifically, o^* is the actual result and o are the returned results, query accuracy is $\frac{\mathrm{DistST}(q,o)}{\mathrm{DistST}(q,o^*)}$.

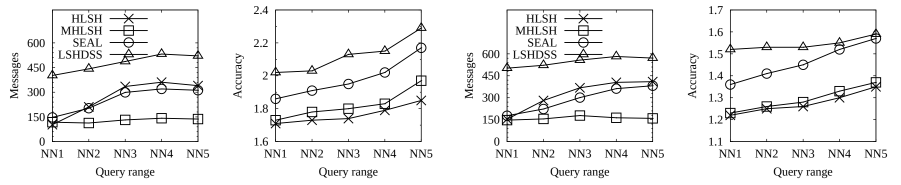
Effect of c. We first investigate the effect of the approximate ratio c on space consumption and query accuracy. Table 1 shows the performance of adaptive hybrid-LSH when $c = 2$ or $c = 3$. Because L is in direct proportion to dataset size, and index size is in direct proportion to L, the index size of MicroBlog is smaller than SynSet. As shown in table 1, the query accuracy of case $c = 3$ is a bit bigger than the case that of case $c = 2$, but it is still good, while the number of messages of case $c = 3$ is about $\frac{1}{3}$ of that of case $c = 2$. So it is worth trading a little accuracy for much higher query efficiency.

Table 1. Effect of c

HLSH	$c = 2$			$c = 3$		
	messages	accuracy	index size	messages	accuracy	index size
MicroBlog	608	1.52	1.18G	196	1.64	229M
SynSet	714	1.63	2.1G	216	1.72	575M

Performance of (D, R, c)-NN query. We now show the results obtained from the (D, R, c)-NN query, Figures 3 show the number of messages and accuracy for $(1, 0.02)$-NN, $(1.5, 0.04)$-NN, $(2, 0.06)$-NN, $(2.5, 0.08)$-NN and $(3, 0.1)$-NN queries on MicroBlog and SynSet. Due to the limited space, in figures $(1, 0.02)$-NN, $(1.5, 0.04)$-NN, $(2, 0.06)$-NN, $(2.5, 0.08)$-NN and $(3, 0.1)$-NN are denoted as NN1, NN2, NN3, NN4 and NN5 respectively. As the distance and range increase, the number of messages cost of HLSH and SEAL increase notably at first and then decrease. When the distance and range increase for HLSH, some (D, R, c)-NN queries LSH-transforms are generated, which result in the increasing of the number of messages, and for SEAL, the efficiency of filtering of the algorithm degrade and it needs check more inverted lists and candidates. However, from $(2.5, 0.08)$ the objects which satisfy the queries increase, in turn the algorithms can terminate earlier and the number of disk accesses decreases. In contrast, the number of messages of MHLSH is stable. The number of messages cost of MHLSH is lower than that of SEAL, which is further lower than LSHDSS. From Figures 3(b) and 3(d), the average accuracy of HLSH is best, and the accuracy of MHLSH is better than that of SEAL.

Figures 4(a) and 4(b) illustrate the trends of number of messages and accuracy of the methods in terms of data size on SynSet. As the dataset size increases, the number of messages of SEAL increases fast, as well as that of

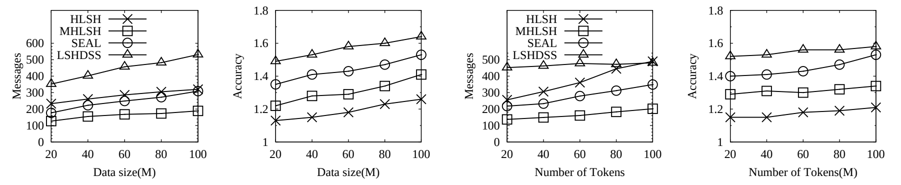


(a) Messages vs. (D, R):MicroBlog (b) Accuracy vs. (D, R):MicroBlog (c) Messages vs. (D, R):SynSet (d) Accuracy vs. (D, R):SynSet

Fig. 3. (D, R, c)-NN with varying (D, R)(Figures share the same key.)

LSHDSS. The reason is that, the number of objects encountered in an inverted list for SEAL is linearly proportional to the dataset size. As a result of two phases of similarity query for LSHDSS, the algorithm is sensitive to the data size. Due to the P_2 of Theorem 1, HLSH and MHLSH are relatively stable with the data size varying. The number of messages cost of MHLSH is still the lowest. Figure 4(b) shows that the accuracies of HLSH and MHLSH are better than those of SEAL and LSHDSS, and HLSH is slightly better than MHLSH at the cost of much more messages.

Figures 4(c) and 4(d) show the performance of (D, R, c)-NN query with different average number of tokens on SynSet. Obviously the number of tokens is similar to the dimensionality in traditional databases. Generally LSH methods are not sensitive to dimensionality, so it's easy to explain that MHLSH and LSHDSS are relatively not affected by the number of tokens. For HLSH, the number of LSH transform queries increases when the number of tokens increases, in result the number of messages of HLSH increases fast. For a fixed query distance and query range, SEAL is in direct proportion to the number of tokens. That's why the number of messages for SEAL increases as the number of tokens increases. Shown in Figure 4(d), the accuracy of HLSH is still the best and the accuracy of MHLSH is better than that of SEAL and that of LSHDSS.



(a) Messages vs. data size (b) Accuracy vs. data size (c) Messages vs. token number (d) Accuracy vs. token number

Fig. 4. (D, R, c)-NN on SynSet(Figures share the same key.)

6 Conclusion

In this paper, we propose a hybrid-LSH scheme for the spatio-textual similarity query. We devise efficient adaptive (D, R, c)-NN algorithm and approximate k-NN method based on the hybrid-LSH scheme. Our theoretical studies show that the algorithms can have a guarantee on query quality. Results of empirical studies demonstrate that the paper's proposal offers scalability and efficiency.

Acknowledgments. This work is supported by the National Basic Research 973 Program of China under Grant (2012CB316201) and the National Natural Science Foundation of China under Grant (61472070). The first author is funded by China Scholarship Council. The third author is partially supported by the grants from NSF NetSE, SaTC, I/UCRC and Intel ICST on Cloud Computing.

References

1. Zhisheng Li, Ken C. K. Lee, Baihua Zheng, Wang-Chien Lee, Dik Lee, and Xufa Wang. Ir-tree: An efficient index for geographic document search. *IEEE Trans. on Knowl. and Data Eng.*, 23(4):585–599, April 2011.
2. Lisi Chen, Gao Cong, and Xin Cao. An efficient query indexing mechanism for filtering geo-textual data. In *Proceedings of the 2013 international conference on Management of data*, pages 749–760. ACM, 2013.
3. Jiaheng Lu, Ying Lu, and Gao Cong. Reverse spatial and textual k nearest neighbor search. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 349–360. ACM, 2011.
4. Ju Fan, Guoliang Li, Lizhu Zhou, Shanshan Chen, and Jun Hu. Seal: Spatio-textual similarity search. *Proceedings of the VLDB Endowment*, 5(9):824–835, 2012.
5. Michael Levandowsky and David Winter. Distance between sets. *Nature*, 234(5323):34–35, 1971.
6. Junhao Gan, Jianlin Feng, Qiong Fang, and Wilfred Ng. Locality-sensitive hashing scheme based on dynamic collision counting. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 541–552. ACM, 2012.
7. Andrei Z Broder, Steven C Glassman, Mark S Manasse, and Geoffrey Zweig. Syntactic clustering of the web. *Computer Networks and ISDN Systems*, 29(8):1157–1166, 1997.
8. Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the 20th annual symposium on Computational geometry*, pages 253–262. ACM, 2004.
9. Xin Cao, Lisi Chen, Gao Cong, and Xiaokui Xiao. Keyword-aware optimal route search. *Proceedings of the VLDB Endowment*, 5(11):1136–1147, 2012.
10. Lisi Chen, Gao Cong, Christian S Jensen, and Dingming Wu. Spatial keyword query processing: an experimental evaluation. In *Proceedings of the 39th international conference on VLDB*, pages 217–228. VLDB Endowment, 2013.
11. Gao Cong, Christian S Jensen, and Dingming Wu. Efficient retrieval of the top-k most relevant spatial web objects. *Proceedings of the VLDB Endowment*, 2(1):337–348, 2009.
12. Parisa Haghani, Sebastian Michel, and Karl Aberer. Distributed similarity search in high dimensions using locality sensitive hashing. In *Proceedings of the 12th International Conference on EDBT*, pages 744–755. ACM, 2009.