Top-k Similarity Join over Multi-valued Objects

Wenjie Zhang, Jing Xu, Xin Liang, Ying Zhang, and Xuemin Lin

University of New South Wales, Sydney, NSW, Australia
zhangw, xjing, xinliang, yingz, lxue @ cse.unsw.edu.au

Abstract. The top-k similarity joins have been extensively studied and used in a wide spectrum of applications such as information retrieval, decision making, spatial data analysis and data mining. Given two sets of objects $\mathcal{U}$ and $\mathcal{V}$, a top-k similarity join returns k pairs of most *similar* objects from $\mathcal{U} \times \mathcal{V}$. In the conventional model of top-k similarity join processing, an object is usually regarded as a point in a multi-dimensional space and the similarity between two objects is usually measured by distance metrics such as Euclidean distance. However, in many applications an object may be described by multiple values (instances) and the conventional model is not applicable since it does not address the distributions of object instances. In this paper, we study top-k similarity join queries over multi-valued objects. We apply quantile based distance to explore the relative instance distribution among the multiple instances of objects. Efficient and effective techniques to process top-k similarity joins over multi-valued objects are developed following a filtering-refinement framework. Novel distance, statistic and weight based pruning techniques are proposed. Comprehensive experiments on both real and synthetic datasets demonstrate the efficiency and effectiveness of our techniques.

1 Introduction

Given two sets of objects (points) $\mathcal{U}$ and $\mathcal{V}$ in a d-dimensional metric space, the top-k similarity join query retrieves k pairs of objects $\mathcal{P}$ from $\mathcal{U} \times \mathcal{V}$ such that the distance between any pair of objects in $\mathcal{P}$ is not greater than the distance of any object pairs in $\mathcal{U} \times \mathcal{V} - \mathcal{P}$. Conventional similarity join query has been extensively studied in various applications including data mining, information retrieval, and location based services [2], [9], [10]. Top-k similarity join, also called *closest pair queries*, has also attracted much research attention [6]. In many applications such as decision making and e-business, an object may be represented by multiple points (instances) in the d-dimensional space, namely *multi-valued* objects [7]. In this paper, we study the problem of top-k similarity joins on *multi-valued* objects.

The needs of similarity join over multi-valued objects stem from many important applications. In geographic information system (GIS), a group of simple spatial objects may be evaluated as a whole [17]. For instance, to evaluate a community, a real estate development company may model it as a multi-valued object and each instance corresponds to a property with some feature values such as property price, household income, distance to beach, distances to living facilities, etc. A top-k similarity join may be issued to identify the most similar communities from two large cities or from two countries, such that the price fluctuation of one community could be used as a mirror to the management of another one. Similarly, in sports, the performance of

a player may be described by her game-to-game statistics in various games. So each player could be represented by a multi-valued object where each instance corresponds to her statistics, such as heights and number of trials in high-jump, in a particular game she attended. A similarity join over two sets of players may help to retrieve players with similar performances. Hence, the successful career path of one player give a prediction of the success of her counterpart in coming competitions.

While the similarity between two conventional d-dimensional objects only involves two single points, identifying the most similar object pairs among multi-valued object sets involves multiple instances per object. Therefore, it is highly desirable to consider the relative instance distributions among multi-valued objects so that the similar pairs can be effectively retrieved. In this paper, we investigate the problem of similarity join over multi-valued objects in a top-k fashion. That is, we aim to retrieve k pairs of multi-valued objects with the highest level of similarity.

The existing model for handling similarity joins over objects with multiple instances follows the probabilistic semantics on uncertain objects [4], [11], [14] and aims to capture relative instance distribution among objects with multiple instances. Nevertheless, uncertain objects are inherently different than multi-valued objects. Instances of an uncertain object are mutually exclusive which means at most one instance can appear at a particular time, while all the values/instances of a multi-valued object must occur simultaneously at any time. Moreover, as shown in [21], models based on uncertain semantics cannot always capture the relative distributions of multi-valued objects. Take the example in Figure 1. For simplicity we assume multi-valued object U_1 has only one instance with the value (score) of 10, while multi-valued objects V_1 and V_2 both have m instances spread between 9.0 to 9.99 as depicted in Figure 1(a). Each instance from the same object takes the same weight. Suppose we want to retrieve the top-1 similarity join result from $\{U_1\}$ and $\{V_1, V_2\}$, namely, retrieve the more similar one from V_1 and V_2 to U_1. Following the possible world semantics, it is easy to verify that both V_1 and V_2 have the same probability, $\frac{1}{2}$, to be the most similar one to U_1 if Euclidean distance is used as the similarity metrics. We permute the distribution in Figure 1(a) to the distribution in Figure 1(b), V_1 and V_2 still have the same probability. This example demonstrates that the probabilistic approaches following the possible world semantics are not able to capture the relative distributions of instances. Another simple solution is to utilize simple aggregates such as average. Nevertheless, such a simple aggregate will have the same problem as pointed above regarding Figure 1.

The example in Figure 1 demonstrates that the existing probabilistic model and simple aggregates may be insensitive to relative distributions of object instances. Quantiles [19] provide a succinct summary of data distributions. In this paper, we investigate the top-k similarity join problem over multi-valued objects based on a ϕ-quantile distance ($\phi \in (0, 1]$); for example, median is the 0.5-quantile, maximum is the 1-quantile, minimum is the smallest quantile (note a quantile ϕ is in $(0, 1]$ and cannot be 0). Regarding the above example, 0.5-quantile is based on players' median performance; 1-quantile is to retrieve the top-k similar pairs based on players' worst performance. In this paper, we study the problem of top-k similarity joins over multi-valued objects where the input are two sets of multi-valued objects.

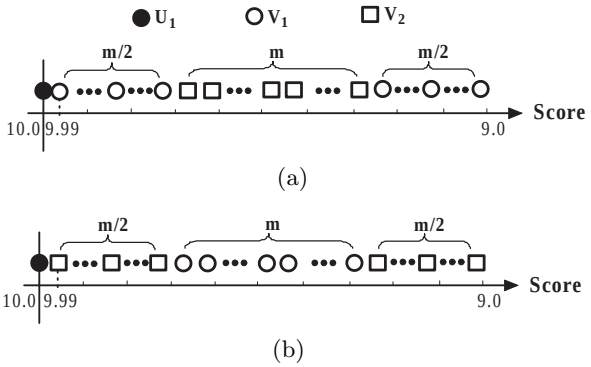


Fig. 1. Motivating Example

Challenges and Contributions. To the best of our knowledge, this is the first paper to study top-k similarity joins over multi-valued objects. ϕ-quantile distance is first used for capturing instance distributions of multi-valued objects in [21]. [21] studies top-k nearest neighbor (KNN) queries over multi-valued objects. Given a multi-valued query object Q and a set of multi-valued objects $\mathcal{U}$, a KNN query retrieve k objects from $\mathcal{U}$ with smallest quantile-based distance to Q. An immediate way to solve our problem can be conducted as follows. For each object $U \in \mathcal{U}$ (or $V \in \mathcal{V}$), we compute its KNN in $\mathcal{V}$ (or $\mathcal{U}$) using the techniques in [21], and then select k most similar pairs based on the union of KNN results. Nevertheless, this involves the computation of KNN for each object in $\mathcal{U}$ (or each object in $\mathcal{V}$). Clearly, not every object in $\mathcal{U}$ (or $\mathcal{V}$) will be involved in the top-k pairs since k is usually much smaller than $min\{|\mathcal{U}|, |\mathcal{V}|\}$. Motivated by this, in this paper, we present a set of novel, efficient, effective pruning techniques to prevent such redundant computation. Our main contributions of the paper can be summarized as follows.

- We formalize the problem of top-k similarity join over multi-valued objects, regarding quantile-based distance metrics.
- Efficient and effective algorithms are developed to compute the top-k similarity join results over two sets of multi-valued objects based on ϕ-quantile-based distances. Particularly, we propose novel and efficient distance, statistic and weight based pruning techniques to significantly speed up the computation.
- Comprehensive experiments are conducted on both real and synthetic data to demonstrate the efficiency and effectiveness of our techniques. It also demonstrates that the techniques developed in this paper are up to 2 orders of magnitude more efficient than naively applying KNN techniques in [21].

Organization of the Paper. The rest of the paper is organized as follows. Section 2 formally defines the problem of top-k similarity join over multi-valued objects regarding quantile-based distance and provide some necessary background information. In Section 3, we introduce the filtering-refinement framework, as well as the data structures utilized in the paper. Section 4 presents query processing techniques for top-k similarity joins. In Section 5, we report our experiment results. Related work is summarized in Section 6. This is followed by conclusions in Section 7.

2 Background

We present problem definition and necessary preliminaries in this section. For references, notations frequently used in the paper are summarized in Table 1.

Notation	Definition
$\mathcal{U}$, $\mathcal{V}$	two sets of of objects in the join query
U (V)	multi-valued object
E	entry of R-tree
u (v)	instance of U (V) - a point in d-dimensional space
$w(u)$ ($w(S)$)	(total) weight of u (the set S)
$d(u, v)$	Euclidean distance between u and v
$d^{lo}(E, E')$	distance lower-bound between E and E'
$d_\phi(U, V)$	ϕ-quantile distance of U and V
$U \times V$	Cartesian product of instances from U to V

Table 1. The summary of Notations.

2.1 Problem Definition

Multi-valued Object. In our problem definition, an instance of an object U is weighted - weight gives the *representativeness* of an instance in U. For instance, in the examples in Section 1, a game statistic of a player may appear multiple times; consequently a normalized weight (the occurrence of an instance over the total occurrences of all instances) may be used to indicate the representativeness of an instance. Note that the total of such weights in U equals to 1.

A multi-valued object U is represented as $\{(u_i, w(u_i))|1 \leq i \leq m\}$ where u_i is a point in a d-dimensional space, $0 < w(u_i) \leq 1$ ($1 \leq i \leq m$), and $\sum_{i=1}^{m} w(u_i) = 1$. We use $\mathcal{U}$ and $\mathcal{V}$ to denote two sets of multi-valued objects involved in the join query.

Quantile. Given a collection S of m elements, each element s_i has a weight $w(s_i)$ where $0 < w(s_i) \leq 1$ and $\sum_{i=1}^{m} w(s_i) = 1$. Let S be sorted increasingly on a search key f - a function; that is, $f(s_i) \leq f(s_j)$ if $i < j$.

Definition 1 (ϕ-quantile of S). *Given a ϕ ($0 < \phi \leq 1$), the ϕ-quantile S_ϕ of S is the* **first** *element s_i in the* **sorted** *S on the search key such that $\sum_{j=1}^{i} w(s_j) \geq \phi$.*

ϕ-quantile Distance. For two given objects U and V, there are totally ($|U| \times |V|$) pairs of instances in $U \times V$ where each pair (u_i, v_j) ($u_i \in U$ and $v_j \in V$) has the weight $w(u_i) \times w(v_j)$, namely $w(u_i, v_j)$. Clearly, $\sum_{u_i \in U, v_j \in V} w(u_i) \times w(v_j) = 1$. The *Euclidean distance* $d(u_i, v_j)$ [1] between u_i and v_j is called the distance of (u_i, v_j). Let $U \times V = \{((u_i, v_j), w(u_i, v_j)) \mid u_i \in U \ \& \ v_j \in V\}$.

Definition 2 (ϕ-quantile distance of U and V). *Given a $\phi \in (0, 1]$, let $U \times V$ be sorted increasingly on the search key - the distance $d(u_i, v_j)$ of each element (u_i, v_j). Then, the distance of the ϕ-quantile of $U \times V$ is called the* ϕ-quantile distance *of $U \times V$, denoted by $d_\phi(U, V)$.*

[1] Note that our techniques developed in this paper is based on Euclidean distance; nevertheless they can be immediately extended to cover other distance metrics.

Definition 2 states that if (u, v) is the ϕ-quantile of $U \times V$ (i.e., $(U \times V)_\phi = (u, v)$) then $d(u, v)$ is $d_\phi(U, V)$.

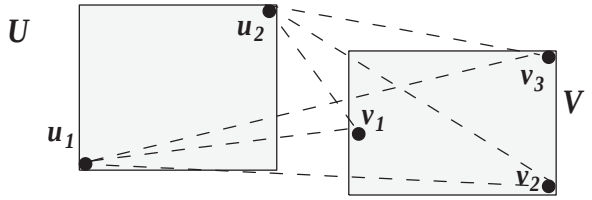


Fig. 2. Distances between 2 Multi-Valued Objects

Example 1. Regarding the example in Figure 2, $|U| = 2$ and $|V| = 3$. Assume that $w(u_1) = w(u_2) = \frac{1}{2}$; $w(v_1) = w(v_2) = \frac{1}{4}$, $w(v_3) = \frac{1}{2}$. Consequently, $U \times V$ consists of the following six instance pairs sorted on their distances increasingly: $U \times V = \{((u_2, v_1),\ \frac{1}{8}),\ ((u_2, v_3),\ \frac{1}{4}),\ ((u_1, v_1),\ \frac{1}{8}),\ ((u_2, v_2),\ \frac{1}{8}),\ ((u_1, v_2),\ \frac{1}{8}),\ ((u_1, v_3),\ \frac{1}{4})\}$. The 0.2-quantile distance $d_{0.2}(U, V)$ of U and V is $d(u_2, v_3)$, $d_{0.5}(U, V)$ is $d(u_1, v_1)$, $d_{0.6}(U, V)$ is $d(u_2, v_2)$. □

Problem Statement. Given a $\phi \in (0, 1]$, two sets of multi-valued objects $\mathcal{U}$ and $\mathcal{V}$ in the d-dimensional space, a top-k similarity join retrieves k pairs of objects $\mathcal{P}$ from $\mathcal{U} \times \mathcal{V}$ such that for each object pair (U, V) from $\mathcal{P}$, its ϕ-quantile distance $d_\phi(U, V)$ is no greater that the ϕ-quantile distance of object pairs from $\mathcal{U} \times \mathcal{V} - \mathcal{P}$.

2.2 Preliminaries

ϕ-quantile Distance Computation. Given a collection S of m elements, each element s_i has a weight $w(s_i)$ where $0 < w(s_i) \leq 1$ and $\sum_{i=1}^{m} w(s_i) \leq 1$. A naive way to compute the ϕ-quantile is to firstly sort S regarding a given search key f, and then scan the sorted list to obtain the ϕ-quantile of S. Clearly, the naive algorithm runs in $O(m \log m)$. In [5], an efficient and effective partitioning technique is proposed to find an element $s \in S$ to divide S into two sub-collections S_1 and S_2 with the following properties:

1. for each $s' \in S_1$, $f(s') \leq f(s)$; and for each $s' \in S_2$, $f(s') \geq f(s)$.
2. $|S_1| \geq \frac{3}{10}m - 6$ and $|S_2| \geq \frac{3}{10}m - 6$.

Using the partitioning technique, when S is not sorted the time complexity of computing ϕ-quantile of S is linear - $O(|S|)$.

Regarding two multi-valued objects U and V, there are totally $|U| \times |V|$ instance pairs. Directly applying the partition based algorithm, computing ϕ-quantile distance between U and V takes $O(|U| \times |V|)$. In [21], instances inside one multi-valued object are indexed by an R-tree. Based on the R-tree, pruning techniques are proposed to discard instance pairs which are guaranteed not to be the ϕ-quantile of $U \times V$. In this paper, we use the pruning techniques enhanced, partition based, linear time complexity algorithm in [21] as a black box in computing ϕ-quantile distance between two multi-valued objects.

Conventional Top-k Similarity Joins. Conventional top-k similarity joins, also called closest pair queries, have been extensively studied over conventional (point) spatial databases [3], [6], [9]. The most recent technique proposes to build an index on the fly [3]. Nevertheless, this technique cannot be used to prune a group of object pairs. That is, every object has to participate in the distance computation. As the quantile distance computation between two objects is very expensive with the presence of multiple instances, in this paper, we will apply an R-tree index based top-k similarity join algorithm to facilitate the prevention of computing quantile distances between unpromising pairs of multi-valued objects. In [6], several algorithms are proposed using R-tree based indexes including exhaustive algorithm, recursive algorithm and Heap algorithm. Among all techniques, Heap algorithm demonstrates a better performance in most experiment settings. The priority query based algorithm in [9] is quite similar to Heap algorithm except that Heap algorithm performs a simple pruning before inserting an entry pair into the heap. We adopt the Heap algorithm and develop novel pruning techniques to speed up the computation. Note that our pruning techniques are general enough to be plugged into any R-tree based algorithm for computing conventional top-k similarity joins.

3 Framework

Our techniques for solving the top-k similarity join based on quantile distance follow a standard seeding-filtering-refinement framework outlined in Algorithm 1.

Algorithm 1: Framework

- **Phase 1 - Seeding:** Compute the ϕ-quantile distance for each of the k chosen object pairs from $\mathcal{U} \times \mathcal{V}$.
- **Phase 2 - Filtering:** Discard unpromising object pairs from $\mathcal{U} \times \mathcal{V}$.
- **Phase 3 - Refinement:** Determine the final solution for ϕ-quantile top-k similarity join.

In the seeding phase, we choose k object pairs and compute their ϕ-quantile distances, using the techniques introduced in Section 2.2. Let λ_k be the maximal of these k ϕ-quantile distances, in the filtering phase, λ_k could be used to prune unpromising object pairs and iteratively updated if necessary. Any k object pairs from $\mathcal{U} \times \mathcal{V}$ could be chosen to compute the ϕ-quantile distance in the seeding phase. Apparently, similar object pairs will lead to smaller λ_k values; and hence better pruning power in the filtering phase. In our framework, to select k object pairs, we first use the mean $\mu(U)$ of the multiple instances for each multi-valued object U from the two given datasets to represent U. $\mu(U) = \sum_{i=1}^{m} w(u_i) \times u_i$ where m is the number of instances in U. Clearly $\mu(U)$ is also in the d-dimensional space. Thus the top-k similarity join is converted to join over conventional datasets where each object is a single point in the multi-dimensional space, and we could apply the existing algorithms [6] to obtain the k most similar pairs from the two (single-valued) datasets. The corresponding k multi-valued object pairs from $\mathcal{U}$ and $\mathcal{V}$ are then chosen to compute the ϕ-quantile distances. At this point, we obtain a distance threshold λ_k which will be used in the filtering phase.

Data Structures

In our techniques, we use aggregate R-trees [16] to index the local instances of each multi-valued object in $\mathcal{U} \cup \mathcal{V}$, and use two statistic information enhanced R-trees (named sR-trees) to globally index the minimum bounding boxes (MBBs) of objects in $\mathcal{U}$ and $\mathcal{V}$, respectively. The local aR-trees and global sR-tress are built to facilitate our filtering techniques.

Local aR-trees. For each multi-valued object $U \in \mathcal{U} \cup \mathcal{V}$, a *local* aR-tree [16] is built to organize its multiple instances. The aggregate information kept on each intermediate entry is the sum of weights of instances indexed by the entry. Namely, for every intermediate entry E in the local aR-tree, we record the weight of E as the sum of weights (total weights) of instances having E as an ancestor.

Global sR-trees. We maintain two R-trees on the MBBs of multiple instances of objects in $\mathcal{U}$ and $\mathcal{V}$, respectively. That is, for each object in $\mathcal{U}$, we first obtain the MBB of its multiple instances. Then we build an R-tree on these MBBs. This R-tree is called the *global* R-tree of $\mathcal{U}$. Similarly we build the *global* R-tree for $\mathcal{V}$. Note in a global R-tree, each leaf (data) entry is an MBB of an object.

Suppose an object U has m instances in the d-dimensional space, $u_1, u_2, ..., u_m$ with the weights $w(u_1), w(u_2), ..., w(u_m)$, respectively.

Definition 3 (Mean μ). *The mean of U, denoted by $\mu(U)$, is $\sum_{i=1}^{m} w(u_i) \times u_i$.*

Note that $\mu(U)$ is in the d-dimensional space. For $1 \leq i \leq d$, $\mu_i(U)$ denotes the i-th coordinate of $\mu(U)$.

Definition 4 (Variance σ^2). *For $1 \leq i \leq d$, $\sigma^2(U) = \sum_{j=1}^{m} w(u_j)(u_{j,i} - \mu_i(U))^2$ where each $u_{j,i}$ denotes the i-th coordinate value of u_j.*

In each of the leaf (data) entry of the global R-tree, besides the MBB information of each object, we also keep the above statistic information. And the global R-tree is called a statistic R-tree, denoted by sR-tree. Remind that two sR-tree are built for the multi-valued object sets $\mathcal{U}$ and $\mathcal{V}$, respectively.

4 ϕ-Quantile Top-k Similarity Join

We present our techniques for ϕ-quantile top-k similarity join for a given $\phi \in (0, 1]$ in this section. We first present novel distance, statistic and weight based pruning techniques. Then, we integrate the proposed pruning techniques into the overall join algorithm based on the Heap Algorithm in [6].

4.1 Pruning Techniques

When introducing the pruning techniques, we assume that we have an entry pair (E_U, E_V) from the join processing where E_U (E_V) is an entry from the global sR-tree of $\mathcal{U}$ ($\mathcal{V}$). E_U (E_V) could be either intermediate or leaf (data) entry. The way to access entries from the two global sR-trees will be introduced in Section 4.2.

Distance based Pruning. The first pruning rule is based on the distance between two entries in the join processing obtained from intermediate or leaf entries of two global sR-trees.

Pruning Rule 1. Let $d^{lo}(E_U, E_V)$ denote the minimum distance between the MBBs of two entries E_U and E_V. If $d^{lo}(E_U, E_V) \geq \lambda_k$, then (E_U, E_V) can be pruned, namely, all entry pairs in $E_U \times E_V$ can be pruned.

Complexity. Computing the minimum distance between two MBBs takes $O(d)$ time. The complexity of Pruning Rule 1 is constant once d is fixed.

Statistic based Pruning. The second pruning technique utilizes the statistic information kept in the global sR-tree, as introduced in Section 3. The main idea is based on the current distance threshold λ_k, to derive a value α such that the α-quantile distance between an object pair (U, V) is not smaller than λ_k. If $\alpha < \phi$, we can safely prune (U, V). We first introduce the *Cantelli's inequality* [15] which is employed in Pruning Rule 2.

Let $\delta(x, y)$ be $\frac{1}{1+\frac{x^2}{y^2}}$ if $y \neq 0$, 1 if $x = 0$ and $y = 0$, and 0 if $x \neq 0$ and $y = 0$.

Theorem 1 (Cantelli's Inequality [15]). *Suppose that t is a random variable in 1-dimensional space with mean $\mu(t)$ and variance $\sigma^2(t)$, $Prob(t - \mu(t) \geq a) \leq \delta(a, \sigma(t))$ for any $a \geq 0$, where $Prob(t - \mu(t) \geq a)$ denotes the probability of $t - \mu(t) \geq a$.*

Note that Theorem 1 extends the original Cantelli's Inequality [15] to cover the case when $\sigma = 0$ and/or $a = 0$. The following theorem is proved in [13] and provides an upper-bound for $Prob(t \leq b)$ when $b \leq \mu$.

Theorem 2. *Assume that $0 \leq b \leq \mu(t)$. Then, $Prob(t \leq b) \leq \delta(\mu(t) - b, \sigma(t))$.*

Proof. Let $t' = 2\mu(t) - t$. It can be immediately verified that $\sigma^2(t') = \sigma^2(t)$ and $\mu(t) = \mu(t')$. Applying Cantelli's Inequality on t', the theorem holds. □

Now we generalize the above observations into our statistic based pruning rule. As shown in Figure 3, for two object entries (U, V) stored in the leaf/data entries of global sR-tree of $\mathcal{U}$ and $\mathcal{V}$, along the i-th dimension ($1 \leq i \leq d$), e.g., the horizontal dimension in Figure 3, we locate two lines m and n vertical to the i-th dimension and with distance λ_k between m and n. Denote U_i (V_i) as the coordinate value of U (V) along the i-th dimension. The line $U_i = m$ ($V_i = n$) divides the MBB of U (V) into two parts, denoted as U_1 and U_2 (V_1 and V_2), as shown in Figure 3. Assume $\mu_i(U) < \mu_i(V)$. Remind that λ_k is the current distance threshold.

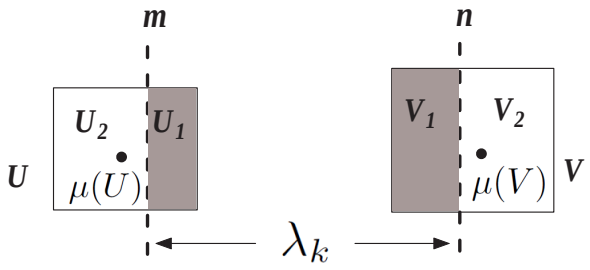


Fig. 3. Statistic based Pruning

The intuition of the statistic based pruning technique is along each dimension i, based on Theorem 2, we derive an upper bound of the sum of weights in the shaded

areas of the MBBs of U_1 and V_1, respectively, denoted as $W_i^{up}(U_1)$ and $W_i^{up}(V_1)$. Clearly, we can claim that instance pairs from $U_2 \times V_2$ can not have distance smaller than λ_k. Denote the sum of weights in U_2 and V_2 as $W_i(U_2)$ and $W_i(V_2)$, respectively. Apparently, $W_i(U_2) \geq 1 - W_i^{up}(U_1)$, and $W_i(V_2) \geq 1 - W_i^{up}(V_1)$. Thus, using $W_i^{up}(U_1)$ and $W_i^{up}(V_1)$, we can identify a value α such that the α-quantile distance between U and V is not smaller than λ_k. Next we present the monotonic property of quantile distance.

Theorem 3 (Monotonicity of Quantile Distance). *Given two multi-valued objects U and V, $\alpha, \phi \in (0, 1]$, if $\alpha < \phi$, then $d_\alpha(U, V) \leq d_\phi(U, V)$.*

Proof. The theorem immediately holds based on the definition of quantile distance in Definition 2. □

Based on Theorem 3, once we identify the value α such that the α-quantile distance between U and V is larger than λ_k, if $\alpha < \phi$, then we can claim the ϕ-quantile distance between U and V cannot be smaller than λ_k. In this way (U, V) can be pruned based on the statistic information kept in the global sR-tree only without accessing the local aR-trees of U and V.

Pruning Rule 2. Given an object pair (U, V) ($U \in \mathcal{U}, V \in \mathcal{V}$). For a dimension i ($1 \leq i \leq d$), without lose of generality, assume $\mu_i(U) < \mu_i(V)$. If 1 - (1 - $\delta(m - \mu_i(U), \sigma_i(U))) \times (1 - \delta(\mu_i(V) - m, \sigma_i(U))) < \phi$, (U, V) can be pruned.

Proof. For the i-th ($1 \leq i \leq d$) dimension, based on Theorem 2, we obtain the upper bound of the sum of weight of instances in the shaded area U_1 of the MBB of U as $W_i^{up}(U_1) = Prob(U_i \geq m) \leq \delta(m - \mu_i(U), \sigma_i(U))$. Similarly we get $W_i^{up}(V_1) = Prob(V_i \leq n) \leq \delta(\mu_i(V) - n, \sigma_i(V))$. Since the instance pairs from $U_2 \times V_2$ cannot have distance smaller than λ_k, we have $\alpha \leq 1 - (1 - W_i^{up}(U_1)) \times (1 - W_i^{up}(V_1)) \leq 1 - (1 - \delta(m - \mu_i(U), \sigma_i(U))) \times (1 - \delta(\mu_i(V) - m, \sigma_i(U)))$. Together with Theorem 3, the pruning rule is correct. □

Once we obtain an object pair (U, V) from the join processing, we apply Pruning Rule 2 based on the statistic information kept in the global sR-trees before accessing the local aR-trees of U and V. If we encounter a dimension i such that 1 - (1 - $W_i^{up}(U_1)) \times (1 - W_i^{up}(V_1)) < \phi$, the pruning rule stops and the object pair (U, V) is discarded. As shown in Figure 3, after selecting line m along the i-th dimension of U, line n for V is also fixed regarding the current λ_k. We apply the equality principle in determining the position of m and n; namely, the center of m and n is the same as the center of $\mu_i(U)$ and $\mu_i(V)$. Based on Theorem 2, we obtain $W_i^{up}(U_1)$ and $W_i^{up}(V_1)$ in constant time.

Complexity. If $W_i^{up}(U_1)$ and $W_i^{up}(V_1)$ are derived based on Theorem 2, the time complexity of Pruning Rule 2 is $O(d)$.

Weight based Pruning. The following pruning rule incorporates both weight and distance information. The instances of a multi-valued object are investigated by accessing the local aR-trees. Consider an object entry pair (U, V). If (U, V) is not pruned by Pruning Rule 1 and 2, we explore the instances information of the objects by accessing their local aR-trees. We traverse the local aR-trees of two objects U and V synchronously. At level i, we *trim* object V using the current distance threshold λ_k,

and retain only the entries in V with minimum distance to U not larger than λ_k. We record the entries as $\gamma_{V,i}$. Formally, $\gamma_{V,i} = \{E \in L_i(V), d^{lo}(U,E) \leq \lambda_k\}$, where $L_i(V)$ denotes all remaining entries (i.e., not trimed in higher levels) in the local aR-tree of V at the i-th level. Similarly, we obtain $\gamma_{U,i}$. If the multiplication of the weights of $\gamma_{V,i}$ and $\gamma_{U,i}$ is smaller than ϕ, the object pair (U,V) can be pruned as the ϕ-quantile distance between U and V must be larger than λ_k.

Pruning Rule 3. If $\sum_{e\in\gamma_{U,i}} W(e) \times \sum_{e\in\gamma_{V,i}} W(e) < \phi$, the object pair (U,V) can be discarded.

Proof. From the definition of ϕ-quantile distance, it is immediate that if $\sum_{e\in\gamma_{U,i}} W(e) \times \sum_{e\in\gamma_{V,i}} W(e) < \phi$, then $d_\phi(U,V) > \lambda_k$. □

Example 2. As shown in Figure 4, at the i-th level, the local aR-tree of object U has two entries U_1 and U_2, local aR-tree of V also has two entries V_1 and V_2. The current threshold λ_k is as illustrated. Using λ_k, we *trim* the MBB of V and only entry V_1 has minimum distance to U smaller than λ_k; thus, $\gamma_{V,i} = \{V_1\}$. Similarly, $\gamma_{U,i} = \{U_2\}$. If $W(U_2) \times W(V_1) < \phi$, the object pair (U,V) could be pruned.

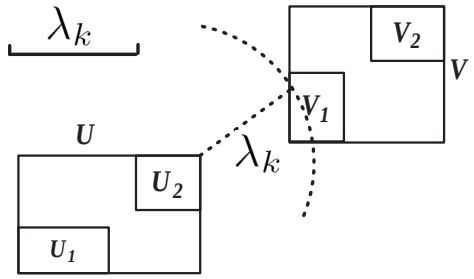


Fig. 4. Weight based Pruning

Applying Pruning Rule 3, we can avoid accessing all instance pairs of $U \times V$, and seek to stop on intermediate levels of the local aR-trees of U and V. Note the traversal of two aR-trees is in a synchronous fashion and level-by-level from the root node. If one aR-tree reaches leaf nodes first, it stays in leaf level while the other one keeps traversing till its leaf level. As a by-product, if (U,V) cannot be pruned using Pruning Rule 3, we call the ϕ-quantile distance computation algorithm in [21] with the instance pairs from $\gamma_{U,i} \times \gamma_{V,i}$ only where i is the leaf (instance) level. Clearly, the algorithm still outputs correct ϕ-quantile distance as the distance of the pruned instance pairs are larger than λ_k based on the definition of $\gamma_{U,i}$ and $\gamma_{V,i}$ for level i.

An exceptional case of Pruning Rule 3 is that we obtain an entry pair (E_U, E_V) sfrom the join processing, one is an object entry while the other is an intermediate entry. Assume E_U is the object entry of U and E_V is the intermediate entry. Pruning Rule 3 could still be applied to (U, E_V) with the following modifications: 1) We access the local aR-tree of U only and at each level i, record $\gamma_{U,i}$ as the entries in U with minimum distance to E_V not larger than λ_k; 2) if $\sum_{e\in\gamma_{U,i}} W(e) < \phi$, the entry pair (U, E_V) could be pruned. Namely, the object pair of U and any object indexed in E_V must have a ϕ-quantile distance greater than λ_k.

Complexity. Assume the average number of entries at level i of the local aR-trees of multi-valued objects is N_i, then clearly the complexity of Pruning Rule 3 is $O(N_i)$ at

each level. The worst case complexity of using Pruning Rule 3 is $O(|U| \times |V|)$, namely no entries are pruned at intermediate entries and we need to access all instance pairs. However, in practice, as shown in Section 5, Pruning Rule 3 is very effective and saves CPU costs significantly. Note that in Pruning Rule 3 we trim the entries at each level of local aR-trees of U and V using λ_k instead of considering the combination of all pairs of entries at each level. This is because trim based pruning is more efficient compared with combining all pairs (time complexity $O(N_i^2)$) and also trim based pruning is very effective in practice.

4.2 Overall Join Algorithm

The join algorithm used in this paper is adopted from the Heap Algorithm in [6] as it is both efficient and easy to implement in real applications. We adjust the algorithm to deal with multi-valued objects. Given $\phi \in (0, 1]$, two multi-valued objects sets $\mathcal{U}$ and $\mathcal{V}$, Algorithm 2 illustrates the top-k similarity join processing. A minheap H is maintained according to the minimum distance between two entry pairs of the two global R-trees $R_\mathcal{U}$ and $R_\mathcal{V}$ indexing $\mathcal{U}$ and $\mathcal{V}$, respectively. H is initialized with the pair of root nodes of $R_\mathcal{U}$ and $R_\mathcal{V}$.

Algorithm 2: Top-k Similarity Join Processing

```
   Input  : R_U, R_V, k, φ
   Output : k object pairs from U × V with smallest φ-quantile distances
 1 H = (root(R_U), root(R_U)) if not PRUNED1(root(R_U), root(R_U));
 2 while H is not empty do
 3     (E_U, E_V) = H.top();
 4     H. pop();
 5     if E_U and E_V are both intermediate entries then
 6         for each children pair (C_{E_U}, C_{E_V}) from E_U × E_V do
 7             if not PRUNED1(C_{E_U}, C_{E_V}) then
 8                 insert (C_{E_U}, C_{E_V}) into H;
 9     else if one of E_U and E_V is an object entry then
10         if not PRUNED1(E_U, E_V) and not PRUNED3(E_U, E_V) then
11             Lines 6 - 8;
12     else                                  /* both E_U and E_V are object entries */
13         if not PRUNED1(E_U, E_V) and not PRUNED2(E_U, E_V) AND not
           PRUNED3(E_U, E_V) then
14             Compute φ-quantile distance between E_U and E_V;
15             if d_φ(E_U, E_V) < λ_k then
16                 Update λ_k and current k most similar pairs;
17
```

The algorithm differentiates three cases based on whether the entries are object entries or not. If both are intermediate entries (Line **5**), we expand all the children pairs and insert into heap H the pairs which survive from Pruning Rule 1 (Line **7**). If one of the entries is an intermediate entry and the other is an object entry (Line **9**), Pruning Rule 1 and 3 will be applied first (Line **10**) before expanding the children pairs. We apply all 3 Pruning Rules object pairs (Line **13**), and if survived, the ϕ-

quantile distance is computed; the top-k results and λ_k are updated if necessary. Note that even from the root node pair we only insert entry pairs into H if they are not pruned by Pruning Rule 1, it is still necessary to check Pruning Rule 1 (Line **9** and Line **13**) since the distance threshold λ_k dynamically changes.

Correctness. Based on the correctness of the 3 pruning rules, it can be immediately shown that Algorithm 2 is correct.

Discussions. The techniques proposed in this paper could be immediately extended to support self-join (i.e., we compute top-k similar pairs from one data set $\mathcal{U}$) and threshold base similarity join over multi-valued objects. We omit the details due to space limits.

5 Experiment

We report a thorough performance evaluation on the efficiency and effectiveness of our algorithms. In particular, we implement and evaluate the following techniques.

Top-k Join: Techniques presented in Section 4 to compute top-k similarity join based on ϕ-quantile distance ($\phi \in (0, 1]$), with all 3 pruning techniques.

P12: Techniques in Section 4 but using Pruning Rule 1 and 2 only (i.e., distance and statistic based pruning).

P1: Techniques in Section 4 but using Pruning Rule 1 only (i.e., distance based pruning).

P0: Techniques in Section 4 but without any pruning rule.

KNN: Baseline algorithm by using KNN processing over multi-valued objects in [21]. For each object $U \in \mathcal{U}$, we compute its KNN in $\mathcal{V}$, and then select k most similar pairs based on the union of KNN results.

All algorithms are implemented in C++ and compiled by GNU GCC. Experiments are conducted on PCs with Intel Xeon 2.4GHz dual CPU and 4G memory under Debian Linux. Our experiments are conducted on both real and synthetic datasets.

Real dataset is extracted from NBA players' game-by-game statistics containing 339,721 records of 1,313 players (http://www.nba.com). Each player is treated as a multi-valued object where the statistics (score, assistance, rebound) of a player per game is treated as an instance with the equal weight (normalized).

Synthetic datasets are generated using the methodologies in [1] regarding the following parameters. Dimensionality d varies from 2 to 5 with default value 3. Data domain in each dimension is $[0, 1]$. Average number n of objects in each dataset varies from $5k$ to $15k$ with default value $5k$. Number of instances per object follows a uniform distribution in [1, m] where m varies from 100 to 800 with the default value 200. The value K varies among 5 to 25 with default value 10. The average length of object MBBs varies from 0.01 to 0.05 with default value 0.01. With the default setting, the total number of instances in a dataset is about $500k$.

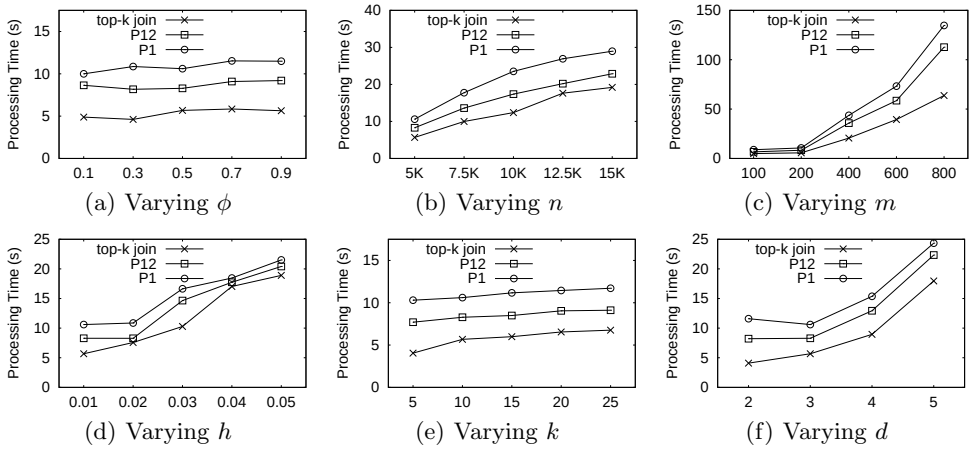
Generating $\mathcal{U}$ and $\mathcal{V}$. In a (real or synthetic) dataset, each object is drawn to $\mathcal{U}$ or $\mathcal{V}$ with equal probability (i.e., probability of $\frac{1}{2}$).

5.1 Overall Performance

Figure 5 reports the results of the evaluation on processing time of Top-k Join, P12, P1, P0, and KNN over real and synthetic data. KNN is very slow and cannot terminate when the dataset size is large, so in the synthetic dataset, we set dataset size to 1k and other parameters take the default values. As shown, each pruning rule is very effective and reduce the processing time significantly. Our techniques are much more efficient compared to a naive application of KNN techniques to processing top-k similarity joins. As P0 and KNN are very inefficient, we omit their performance evaluation in the following sections.

Fig. 5. Overall Performance

5.2 Evaluating Impacts by Different Settings



(a) Varying ϕ (b) Varying n (c) Varying m

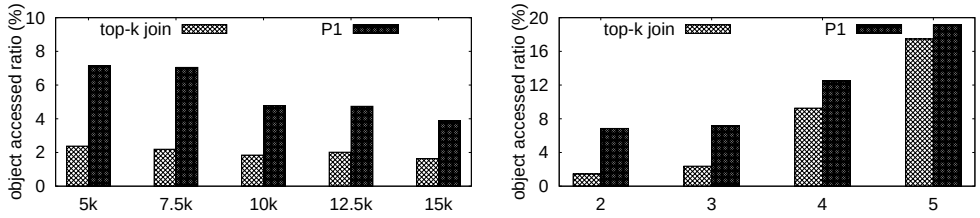
(d) Varying h (e) Varying k (f) Varying d

Fig. 6. Varying Parameters

In this subsection, we study the scalability of our algorithms regarding different ϕ-values, number of objects in one dataset(n), number of instances (m), length of MBB edges (h), k and the dimensionality d in Figure 6. While our techniques are not very sensitive to ϕ-values and k, the processing time increases with the increase of number of objects, instance number, MBB edge length, and dimensionality. Clearly, the dataset size increases with objects and instances number thus the join processing becomes more expensive. Larger MBB edge length makes it difficult to prune object pairs as there is larger overlap in their MBBs. The results also demonstrate that each pruning rule is very effective and significantly reduces the processing time.

5.3 I/O Costs

We report the ratio of objects accessed in Figure 7. Since Pruning Rule 3 mainly works on two object entries and load both two objects, its effect is on saving CPU time but not I/O. Thus, we report the performance of Top-k Join and P1 only. As shown in the figure, our techniques could achieve dramatic saving on the total objects loaded. Compared to P1, the saving from using Pruning Rule 2 is also significant.



(a) Varying n (b) Varying d

Fig. 7. Object Accessed Ratio

6 Related Work

Conventional join queries over two multi-dimensional datasets are fundamental in data analysis and information retrieval. Most existing techniques for join queries have been developed based on popular spatial access methods such as R-trees. For threshold based joins, there are three main stream spatial join algorithms using R*-tree [8]. They are the depth-first-join (DFJ) algorithm [2], the breadth-first-join (BFJ) algorithm [10], and transformation-view-join (TVJ) algorithm [12]. Techniques for top-k spatial/similarity queries are studied in [6], [9]. Various algorithms, such as exhaustive algorithm, recursive algorithm, Heap algorithm, and priority queue based algorithms are proposed. Many variation of join queries over multi-dimensional space have been studied in different contexts, including road networks [18] and moving objects [20].

Spatial queries such as nearest neighbor queries over fuzzy objects have been recently studied [22]. Fuzzy objects possess similar semantics as uncertain objects (e.g., instances are mutually exclusive). The techniques in [22] are not applicable to the problem studied in our paper due to the different semantics as well as inherent difference in query types.

Join queries over uncertain objects are inherently different than conventional joins where each uncertain object takes a set of mutually exclusive instances/points in a multi-dimensional space. It is extensively studied in [4], [11], [14]. Note that all instances in a multi-valued object exist simultaneously instead of mutually exclusive in an uncertain object. Due to such inherent differences in semantics, join techniques over uncertain objects cannot be directly applied to similarity joins over multi-valued objects.

7 Conclusion

We study the problem of top-k similarity join over multi-valued objects. The distance/similarity between two multi-valued objects is measured using quantile based

distance to capture the relative instance distribution. A filtering-refinement framework is developed, along with novel, efficient and efficient distance, statistic and weight based pruning techniques. Comprehensive experimental study over both real and synthetic datasets demonstrates the efficiency and scalability of our techniques.

Acknowledgement. Wenjie Zhang is supported by ARC DP120104168 and ARC DE120102144. Ying Zhang is supported by ARC DP110104880 and UNSW ECR grant PSE1799. Xuemin Lin is supported by ARC DP0987557, ARC DP110102937, ARC DP120104168, NSFC61021004 and a Google Research Award.

References

1. S. Borzsonyi, D. Kossmann, and K. Stocker. The skyline operator. In *ICDE 2001*.
2. T. Brinkhoff, H.-P. Kriegel, and B. Seeger. Efficient processing of spatial joins using r-trees. In *SIGMOD 1993*.
3. M. A. Cheema, X. Lin, H. Wang, J. Wang, and W. Zhang. A unified approach for computing top-k pairs in multidimensional space. In *ICDE 2011*.
4. R. Cheng, S. Singh, S. Prabhakar, R. Shah, J. S. Vitter, and Y. Xia. Efficient join processing over uncertain data. In *CIKM 2006*.
5. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. Introduction to algorithms 2nd edition. chpater 9: Medians and order statistics. In *The MIT Press*.
6. A. Corral, Y. Manolopoulos, Y. Theodoridis, and M. Vassilakopoulos. Closest pair queries in spatial databases. In *SIGMOD 2000*.
7. R. Elmasri and S. Navathe. Fundamentals of database systems. (6th edition). In *2011*.
8. W.-S. Han, J. Kim, B. S. Lee, Y. Tao, R. Rantzau, and V. Markl. Cost-based predictive spatiotemporal join. In *TKDE 2009*.
9. G. Hjaltason and H. Samet. Incremental distance join algorithms for spatial databases. In *SIGMOD 1998*.
10. Y.-W. Huang, J. Ning, and E. A. Rundensteiner. Spatial joins using r-trees: Breadth-first traversal with global optimizations. In *VLDB 1997*.
11. H.-P. Kriegel, P. Kunath, M. Pfeifle, and M. Renz. Probabilistic similarity search on uncertain data. In *DASFAA 2006*.
12. M.-J. Lee, K.-Y. Whang, W.-S. Han, and S. I.-Y. Transform-space view: Performing spatial join in the transform space using original-space indexes. In *TKDE 2006*.
13. X. Lin, Y. Zhang, W. Zhang, and M. A. Cheema. Stochastic skyline operator. In *ICDE 2011*.
14. V. Ljosa and A. K. Singh. Top-k spatial join of probabilistic objects. In *ICDE 2008*.
15. R. Meester. *A Natural Introduction to Probability Theory*. 2004.
16. D. Papadias, P. Kalnis, J. Zhang, and Y. Tao. Efficient OLAP operations in spatial data warehouses. In *SSTD 2001*.
17. P. Rigaux, M. Scholl, and A. Voisard. Spatial databases: With applications to gis. In *2001*.
18. J. Sankaranarayanan, H. Alborzi, and H. Samet. Distance join queries on spatial networks. In *GIS 2006*.
19. M. L. Yiu, N. Mamoulis, and Y. Tao. Efficient quantile retrieval on multi-dimensional data. In *EDBT 2006*.
20. R. Zhang, D. Lin, K. Ramamohanarao, and E. Bertino. Continuous intersection joins over moving objects. In *ICDE 2008*.
21. W. Zhang, X. Lin, M. A. Cheema, Y. Zhang, and W. Wang. Quantile-based knn over multi-valued objects. In *ICDE 2010*.
22. K. Zheng, P. Fung, and X. Zhou. K nearest neighbor search for fuzzy objects. In *SIGMOD 2010*.