Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 5, 2021, Pages. 3918 - 3927
Received 15 April 2021; Accepted 05 May 2021.

User Centric Similarity Search

K.Deepa, Professor/IT,Sri Ramakrishna Engineering College
Anjali J, Student,Sri Ramakrishna Engineering College
Devika RStudent, Sri Ramakrishna Engineering College
Dharshana S S,Student, Sri Ramakrishna Engineering College

Abstract: In market analysis, user priorities are very important. There has been a massive amount of work on query primitives in the database literature, such as the well-known top-k query, which can be used to rank products based on consumer preferences. Nonetheless, the basic operation of evaluating product similarity is often performed without regard for these preferences.Instead, products are represented in a feature space based on their characteristics, and similarity is calculated using conventional distance metrics. In this paper, we use product rankings based on consumer feedback to map products in a user-centric space where similarity calculations are performed. We define some of the mapping's most significant characteristicsthat result in upper and lower similarity boundsin order to perform these user-centric similarity computations, we can use traditional multidimensional indexes on the original product space. We demonstrate how to efficiently perform interesting similarity calculations inspired by the widely used range and nearest neighbor queries, thus pruning significant portions of the databased on the bounds we derive on the user-centric similarity of products.

Index Terms: Similarity Estimation, Reverse top-k query, Top-k query, user preferences

1. INTRODUCTION

The ESTIMATION of object similarity is a basic data processing process. It's used to identify sites or documents on the internet that have similar terms, or to spot consumers that have abnormal behavior based on the items they purchase. Furthermore, similarity computations can be used to detect related conversations and comments between social media users (i.e. comments on Facebook, tweets on Twitter).

Many different similarity metrics, such as the Euclidean distance and the cosine similarity, have been proposed for measuring the similarity between two data objects. Such metrics imply that the correlation between data items is calculated solely on the basis of their characteristics, without taking into account the views of users. In business analysis, for example, goods are represented as points with attributes specified by their values. The more identical two products are according to the chosen metric, the closer they are to each other.

In this paper, we present a user-centric approach to similarity computation that takes users' expectations into account. For example, a business manager may be interested in the effect of its business products on consumers as opposed to existing products from competitors. It's important for her to know which items are on the wish lists of as many different consumers as possible. This information may be used to concentrate on goods with similar categories of consumers who rank them highly based on their preferences. Then a more effective marketing strategy might be implemented, resulting in product clusters that are more appealing to individual consumers.

As a result, a business manager should be able to run a query that returns similar items (even if they have never been rated) based on both their characteristics and the preferences of the customers. She must see the data through the eyes of the users and conduct similarity computations based on the preferences of the users who are available. Users' expectations are represented as vectors of weighting factors for item attributes in our system. We use a query type known as a reverse top-k query to perform similarity computations of this type. A reverse top-k query returns the set of customers for which a given product belongs to their top-k set, as opposed to a top-k query, which returns the k items with the best score for a particular customer. Our work may also be applied to goods that have recently been introduced to the market or are in the design phase of the production process and have yet to receive customer feedback.

We use the Jaccard coefficient to compute similarity between the resulting sets of reverse top-k queries in our research. It is also taken into account an expanded notion of similarity that takes into account the product ranks. These new similarity metrics are supplemented by two new query forms, -similarity and m-nearest neighbour queries, which are similar to the well-known range and nearest neighbour queries but vary in that they measure product similarity by looking at their reverse top-k sets.

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 5, 2021, Pages. 3918 - 3927
Received 15 April 2021; Accepted 05 May 2021.

2. LITERATUREREVIEW

The section provides a review of the literature on Data Mining and other query processing techniques. Data mining, also known as database knowledge discovery, was described in 2011 as "the process of analyzing large data repositories and discovering tacit but potentially useful information" (Han, Kamber, & Pei) [1] In 2006, Data mining has the capability to reveal hidden relationships and to discover the unknown patterns and trends by mining huge amount of data.(Sumathi &Sivanandam) [2]. From 2000 to 2006, data mining functions (or models) were divided into four categories based on the task at hand: association, grouping, clustering, and regression (Hui & Jha, 2000; Kao, Chang, & Lin,2003;Nicholson,2006) (3).

Classical statistics is primarily used to investigate data, data relationships, and numeric data in large databases (David J. Hand, 1998). Regression analysis, cluster analysis, and discriminate analysis are examples of classical statistics. Artificial intelligence (AI) is a statistical problem-solving technique that uses "human-like" computation. Genetic algorithms, fuzzy logic, and neural computation are some of the methods used in AI. . Finally, machine learning is a tool for data processing and information exploration that combines advanced statistical methods with AI heuristics (Kononenko&Kukar, 2007). Neural networks, symbolic learning, genetic algorithms, and swarm optimization are some of the methods used in machine learning. These technologies benefit data mining, but the goal is different: extracting patterns, explaining trends, and predicting behavior. Data mining techniques are used to identify unknown or secret knowledge in a broad variety of domains where vast volumes of data are available. In this context, data mining methods used on the internet are referred to as web mining, text mining is referred to as text mining, and bibliomining is referred to as bibliomining. (2006, N. Girija and S.K. Srivatsa) [4]

In the year 2011, A typical data mining process consists of a series of interactive steps that begin with the integration of raw data from various data sources and formats. These raw data are cleansed to eliminate noise, duplicated data, and inconsistencies. These cleansed data are then converted into suitable formats that other data mining software may understand, and filtration and aggregation techniques are used to extract summarized data. (Han et al.) [5]

In the year 2003, Scott Nicholson and Jeffrey Stanton (2003) coined the term bibliomining, or data mining for libraries, to describe the combination of data warehousing, data mining, and bibliometrics. This term refers to the study of purchase patterns, shifts in behavior, and trends in library systems. While the idea is not new, the word bibliomining was coined to make searching for the term's "library" and "data mining" in libraries rather than software libraries easier. Bibliomining is a valuable method for finding useful library knowledge in historical data and using it to help decision-making (Kao et al). [6].

Vlachou et al. introduced the reverse top-k query and discussed its two types, monochromatic and biochromatic models, in 2004. Vlachou et al. and Chester et al. present several algorithms in a 2-dimensional (2D) space to efficiently address the monochromatic reverse top-k question. Real Time-A and Grand Real Time-A are biochromatic top-k query algorithms (Vlachou et al. and Chester et al.,) [7].

In 2007, a dynamic index was created to help reverse top-k queries, and all top-k queries were used to improve the reverse top-k query. Reverse top-k queries have recently received a lot of attention in the fields of market research, location-based services, and unpredictable situations. It's worth mentioning that all current reverse top-k queries simply return the results with no clarification, which means that existing reverse top-k query techniques can't effectively answer corresponding why-not questions. (Yu et al. AND Ge et al.) [8]

We summarize some representative works here because reverse top-k queries are implicitly related to top-k database processing. Dominating Top-k Aggregate Query, a semantics of ranking queries on unknown big data by integrating skyline and top-k queries, was introduced in 2014 with the aim of providing trustworthy and accurate insightful information. [Cao and Nguyen] [9]

3. PROPOSED SYSTEM

We use the Jaccard coefficient to compute similarity between the resulting sets of reverse top-k queries in our research. It is also taken into account an expanded notion of similarity that takes into account the product ranks. These new similarity metrics are supplemented by two new query forms, -similarity and m-nearest neighbour queries, which are similar to the well-known range and nearest neighbour queries but vary in that they measure product similarity by looking at their reverse top-k sets. Unfortunately, since reverse top-k queries have a high computational expense, brute-force evaluation of these queries is impractical, even for moderate data sets. In this paper, we describe key properties of reverse top-k results that allow us to compute upper and lower bounds on

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 5, 2021, Pages. 3918 - 3927
Received 15 April 2021; Accepted 05 May 2021.

product user-centric similarity. We can efficiently execute these queries using a traditional R-tree index on the product space based on these bounds. Additional optimizations that re-use previously computed reverse top-k sets are also discussed in order to further constrain the similarity of products under investigation. In conclusion, this work makes the contributions that follows

• We present a new paradigm for user-centric similarity search that uses product rankings based on user interests to find related items.

• For user-centric similarity search, we define two new query types (-similarity and m-nearest neighbour), identify effective score bounds, and present efficient query processing algorithms that prune the search space using the derived bounds and conventional index structures.

• We show how our methods can be improved by using a particular similarity measure that captures user-centric similarity in a more fine-grained way.

• We demonstrate how results computed while a database is being processed can be used to derive tighter bounds, significantly improving query processing efficiency.

• We conduct a comprehensive experiment to illustrate the efficiency and efficacy of user-centric similarity search. One of the most important findings of our research is that when we use user expectations instead of conventional similarity computations, we often get very different results. We discovered that in the majority of questions, users prefer the findings obtained using our techniques while conducting studies on two different real-time datasets.

The following are used in this system and all are listed below:

3.1 Similarities And Query

A user-centric approach to similarity computation is presented in this module, in which the similarity of two items is determined by taking into account the user's expectations. When two items satisfy the same user expectations, their reverse top-k sets are considered identical. Assume there is a group of customers C. Each customer ci has a weighting vector wi that she has used to determine customer's preferences. As previously stated, if the query point q is one of the k products returned by the top-k query for customer ci, a weighting vector wi will be part of the result of the reverse top-k collection of a product q. We introduce a metric for the distance (i.e., similarity) of these sets in order to perform similarity computation over the products of RTOPk sets. The Jaccard coefficient is a well-known similarity function for sets of objects. The Jaccard similarity index compares members from two sets to determine which are common and which are unique. It's a percentage measure of how close the two sets of data are, ranging from 0% to 100%. The higher the percentage, the closer the two groups are.

3.2 Processing Of Queries

We explain the indexing structure used for efficient user-centric similarity search in this section, as well as the upper and lower bounds on the similarity of query point q to index entries and the similarity search algorithms.

3.2.1 Indexing

We explain the indexing structure used for efficient user-centric similarity search in this module, as well as the upper and lower bounds on the similarity of a query point q to index entries and the similarity search algorithms. A multidimensional access method is needed to allow efficient access to the product data set. MBRs (Minimum Bounding Rectangles) form a tree hierarchy in the R-Tree, which organizes the products in (hyper)rectangles. Each MBR is described as the smallest rectangle that contains/encloses a collection of products at the lowest (leaf) level, while at higher levels in the tree, multiple MBRs from the next level are grouped together and represented by a new MBR that contains all of the group's rectangles' products.

3.2.2 Bonds Of Similarity

We define upper and lower bounds on the similarity sim(p, q) of q with any product p enclosed in this MBR given a query point q and an MBR M(l, u) identified by corners l and u. (raise a binary vector with the reverse top-k result of point q).

3.2.3 Algorithms

In what follows, query processing algorithms for the evaluation of the two query types introduced. For both queries, we assume that the set of products is indexed using an R tree extended to contain the RTOPk sets of the lower and upper corner of each MBR M (l, u). For efficient computation of the RTOPk sets, we utilize the RTA algorithm.

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 5, 2021, Pages. 3918 - 3927
Received 15 April 2021; Accepted 05 May 2021.

RTA expedites query processing by discarding weighting vectors that cannot belong to the result set of RTOPk(q), without evaluating the corresponding top-k queries. As such, RTA reduces the number of top-k evaluations, based on the observation that top-k queries defined by similar weighting vectors return similar results. For this reason, prior to query processing, the weighting vectors are sorted based on their pairwise similarity. By exploiting this order, RTA avoids processing top-k queries that correspond to subsequent vectors in this order. During processing, a buffer containing the k objects (results) of the previous top-k query that was processed is used to discard subsequent vectors, if the objects in the buffer are ranked higher than q. RTA is not significantly affected when the cardinality of the product space or the value of k increases. On the other hand, in case of increase of the weighting vectors cardinality, the number of top-k evaluations necessary for the computation of the result set of a RTOPk query also increase. In the worst case, the RTA algorithm has to process |W| top-k queries, but in practice the algorithm returns the correct result by evaluating much fewer top k queries than |W|. A lower bound of at least |RTOP k(q)| top k queries should be executed, since no weighting vector can be added in the result set without evaluating the respective top-k query.

3.2.3.1 θ−SIMILARITY QUERIES

The -similarity(q) question, by definition, returns all products p that satisfy the condition sim(p, q). We'll use the R-tree index to respond to this question by recursively expanding its nodes in a top-down manner, expanding a node only if it contains a candidate product for inclusion in the result set.

3.2.3.2 NEAREST NEIGHBOR QUERIES

Our algorithm searches the m products in the R-tree, starting from the root node, to find the m nearest neighbours of a given product q. The R-root tree's is the one who starts the priority queue. The algorithm calls itself recursively as long as the m nearest neighbours have not been identified and L is not zero. The first element M of L is removed each time. If it's a single product, it'll be added to the list of nearest neighbours, which will be returned when the list's size reaches m. Since no other product can have a higher similarity, the sorted access and properties of the upper bounds guarantee that the elements of the nn list are the m nearest neighbours. When the R-tree reaches leaf node M (line 8), a reverse top-k query is run for any product pi that belongs to the same MBR M, and the raspberry pi vector is populated (line 10). The goods are also added to the priority queue L, with an upper bound equal to the query product sim's actual similarity (pi; q).

3.3 EXTENDED JACCARD SIMILARITY

A top-k query's result set is bounded to the k products with the best score, while the reverse top-k result set is unbounded; it can (theoretically) become equal to the number of customers or even be zero. To put it another way, a reverse top-k query for a given product can return no weighting vector (customer). The similarity estimation can be inadequate or impossible in the case of products with limited or empty reverse top-k sets. Increase the value of parameter k to enlarge the size of the reverse top-k sets, which is a workaround for this issue.However, since the Jaccard coefficient does not take into account the actual ranking of the product in the top-k list of a consumer who belongs to the product's RTOPk range, this could result in an incorrect estimate of similarity. As a result, the Jaccard coefficient can consider two products to be identical, even if the first is ranked highly and the second is ranked much lower. As the value of parameter k is increased, such differences become more apparent.

To overcome the Jaccard coefficient's constraint, we propose an expanded schema that takes into account a product's actual ranking role in a customer's top-k collection. In a nutshell, instead of assigning one as the value of coordinate rp[i], we assign a positive value derived from the product's ranking position in the customer's list if customer ci belongs to the reverse top-k range of product p. As a consequence, vectors rp are made up of real positive numbers rather than binary values. In this way, extended Jaccard considers two products that are ranked in similar positions by a group of customers to be more similar than any other product that is ranked very differently by the same customers.

3.4 OPTIMIZATIONS

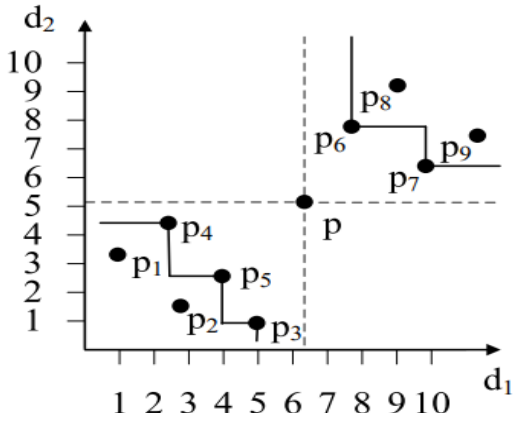
A possible disadvantage of the previously described algorithms is that each time a leaf-level MBR M is visited, a reverse top-k question for each product contained in M must be processed. This induced cost contributes significantly to the algorithms' total processing costs. We present optimizations in this section that aim to reduce the number of reverse top-k computations, resulting in quicker query processing. We use the result sets of previously evaluated reverse top-k queries to get tighter upper and lower similarity limits for (any product p enclosed by) a leaf MBR M and query point q, reducing the number of reverse top-k evaluations. We can prune several points of a

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 5, 2021, Pages. 3918 - 3927
Received 15 April 2021; Accepted 05 May 2021.

given leaf MBR without computing their reverse top-k sets by using the modified bounds, as long as we know they won't be part of the query result.

3.4.1TIGHTER SIMILARITY BOUNDS

The question is how to tighten the bounds of p given a point p to be processed and a set of points (maintained with an in-memory R-tree I) with already computed reverse top-k sets. Graph 1 depicts an illustration that aids in the understanding of this mechanism. Let p1,, pf denote the previously computed points in the two boxes identified by (a) the axes' origin and p (lower box), and (b) p and the data space's maximum corner (upper box). To update the bounds of p, we claim that only a subset of these points is needed. We 1 7 2 4 3 5 6 1 2 9 8 FIG 4. Points required to compute tighter bounds, as an example. Only the points that belong to the skyline in relation to pare needed. Since p dominates this set of points, we call it Domd(p).



Graph 1: Shows an example of necessary points for computing tighter bounds.

Domd(p) = p6, p7 in the diagram. If we consider p to be the space's root, we just need the points that belong to the skyline with respect to p from the lower box. Since these points dominate p, we call them Dom(p). Dom(p) = p3, p4, p5 in the diagram. The union of reverse top-k results of points in Domd and the intersection of reverse top-k results of points in Dom(p) are computed (p). The binary vectors rDom(p) and rDomd(p) are denoted by rDom(p) and rDomd(p), respectively. As stated by the following lemmas, we believe that these vectors can be used to compute tighter bounds for p. (If one of these sets is empty, the corresponding bound remains unchanged.)

3.4.2 IMPROVED QUERY PROCESSING

We boost the efficiency of our query processing algorithms based on the above by updating the bounds of a point before running the reverse top-k query. In more information, when our algorithms approach a leaf MBR, they must process a series of reverse top-k queries (one for each enclosed point). We retrieve sets Dom(p) and Domd(p) by querying the in-memory R-tree I, compute vectors rDom(p) and rDomd(p), and update the bounds of p as defined above before executing each query for point p. We may also stop running the reverse top-k question so the bounds become tighter.

3.4.2. HANDLING UPDATES

TOPk query for c and update the r vectors of those MBR corners that are affected by the customer's insertion. In more information, the r vectors of MBR corners that are occupiedby the kth product in customer c's favorite list do not alter. As a result, we test an RTOPk question for those R-tree MBR corners that are not dominated by the kth product in customer c's TOPk result collection. When a customer is deleted, on the other hand, we only set the entries of all the r vectors that apply to the same customer to zero.

When a new product p is applied to the data set P, all non-zero values of the r vectors, which represent MBR corners occupied by p, are increased by one. There is no action for those r vectors that represent the RTOPk set of MBR corners that dominate p. All other r vectors should be recalculated using an RTOPk query for the corner they represent. If a substance p is removed, the procedure is the same as before. The only difference is that the non-zero values of the r vectors, which reflect MBR corners dominated by p, are decreased rather than increased. We may use batch mode

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 5, 2021, Pages. 3918 - 3927
Received 15 April 2021; Accepted 05 May 2021.

insertion to reduce the expense of the RTOPk queries that need to be performed and there are deletions in data set P. We run an RTOPk for all the corners of the new MBRs generated after the merge/split phase if two or more R-tree nodes are merged or if an R-tree node splits.

4 EXPERIMENTAL EVALUATION

The results of two user study tests that we conducted in order to determine the efficacy of our proposed method are described first in this section. The proposed user-centric similarity search is then put to the test in a large-scale experiment. On a variety of datasets, algorithms were tested. The experiments were run on a desktop PC with an i7 CPU (4 cores, 3.4 GHz), 8GB RAM, and a 128GB SSD, and all variants of the presented algorithms were implemented in Java.

4.1 User Studies Evaluation

We started with a small user survey, asking 30 academic users to fill out a questionnaire. We used a basketball dataset that included five annual statistics (points, rebounds, assists, steals, and blocks) NBA players' performances We chose five users and asked each of them to create a weighting vector that reflects their preferences for the NBA players in the center/forward results. We then asked them to repeat the process for players who play the position of point/shooting guard. Then we chose 20 queries/players at random (10 from each party of center/forwards). We used the extended Jaccard metric to run our 1 NN algorithm to find the most similar player for each question (point/shooting guards). Then, each of the 30 users was given the option of choosing between the 1 NN returned by our process and the 1 NN obtained using the Euclidean distance similarity metric. The majority of users choose the player that our method recommended in the majority of cases (13 out of 20 queries). The majority of users chose the player that the method used only for five queries. The similarity metric based on Euclidean distance was returned. The results of both methods are selected by the same number of users for the two remaining queries.

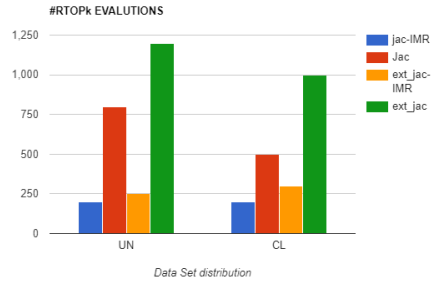
We used the well-known crowdsourcing platform CrowdFlower to launch a task asking 220 contributors/users to answer each of the above questions in order to test this use case scenario on a larger population. The group in this study differs from the previous one in that the initial group of 30 users was chosen based on their experience with the NBA. On the other hand, we had no influence over the contributors who participated in the crowdsourcing platform's user analysis. Even though, the outcomes were very similar. For ten questions, the majority of users chose our method's recommendations, while for one question, the same number of users chose both alternatives. Including the fact that in the remaining 9 questions, users preferred the response suggested by the Euclidean distance, and we found that the differences in user opinions between that choice and our method's recommendation were often small. On the contrary, our approach was favored by 70% or more of users for 6 out of 10 queries. There was only one question where more than 65 percent of users chose the result for the Euclidean distance

4.2 Experiment Setup

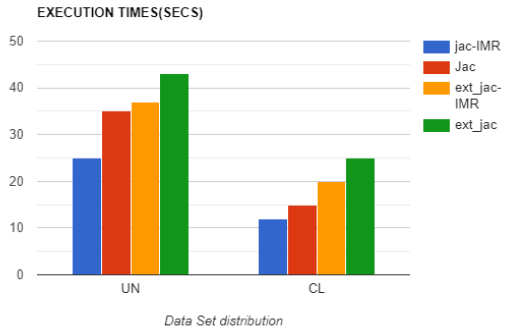
Sets of data we use both actual and synthetic data sets in the experimental analysis. In the case of synthetic data sets, we used uniform (UN) and clustered (CL) distributions to create products.In either case, each attribute's generated values are in the [0, 10K] range. All attribute values in the UN data set are generated independently using a uniform distribution. To create the CL data set, we first create CS cluster centroids, then generate d values (assigned to each of the data vector's coordinates) that follow a normal distribution with variance 2 S and a mean equal to the corresponding centroid coordinate. A disk-resident R-tree with an 8KB block (node) size and a buffer of 100 nodes is used to index the data set P.Additionally, we used four real data sets: NBA, WINE, COLOR, and HOUSE. NBA is made up of 17265 5-dimensional tuples that reflect NBA players' statistics in five different categories. WINE is a UCI Machine Learning 10-dimensional data set. The values of ten characteristics (such as alcohol, magnesium, color strength, and total phenols) of 6497 different types of wines are stored in this database. COLOR is made up of 68040 9-dimensional tuples that describe image features in HSV color space. HOUSE is a six-dimensional data collection that shows how much of 127930 American families' annual income is spent on six different forms of expenses (i.e., gas, electricity, water, heating

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 5, 2021, Pages. 3918 - 3927
Received 15 April 2021; Accepted 05 May 2021.

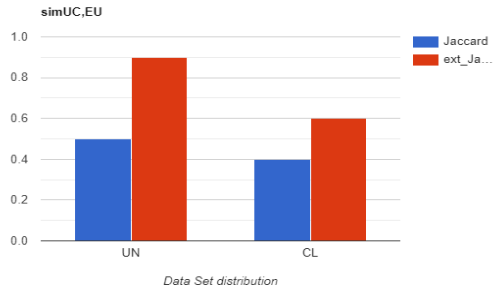
Fig. 2. θ−Similarity Queries Performance



(a)

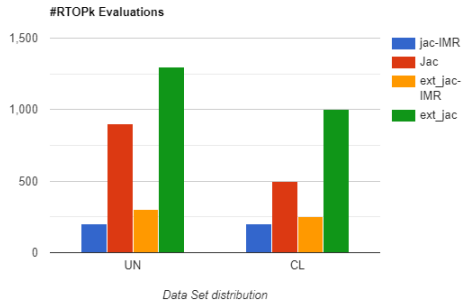


(b)



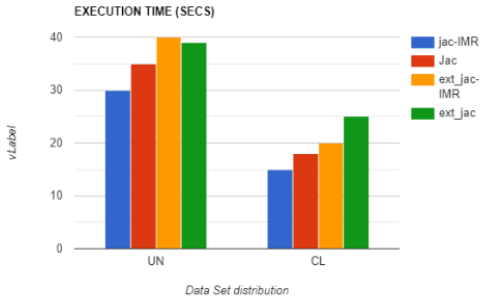
(c)

Fig. 3. m − NN Queries Performance

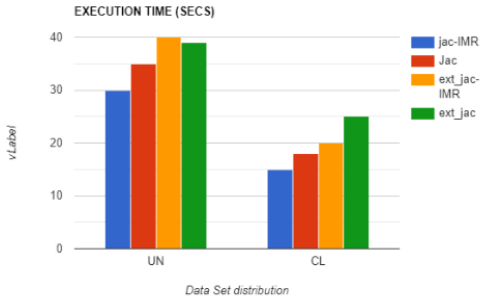


(a)

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 5, 2021, Pages. 3918 - 3927
Received 15 April 2021; Accepted 05 May 2021.

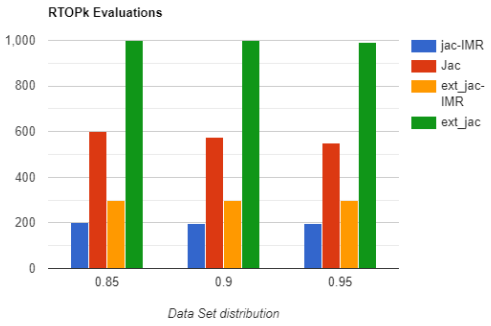


(b)

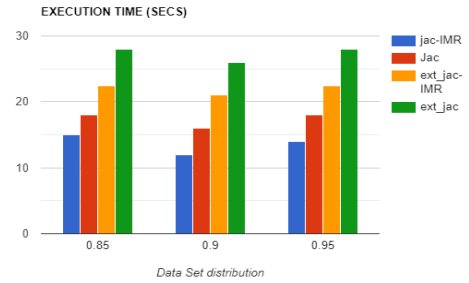


(c)

Fig. 4. θ-Similarity Queries Performance Varying Threshold θ

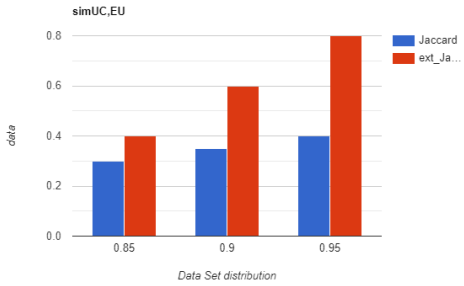


(a)



(b)

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 5, 2021, Pages. 3918 - 3927
Received 15 April 2021; Accepted 05 May 2021.



(c)

We used a clustered (CL) distribution for the weighting vectors data set W. First, CW cluster centroids belonging to the (d-1)-dimensional hyperplane was described and chosen at random by Pwi = 1. Then, on the (d-1)-dimensional hyperplane, each coordinate is created by following a normal distribution on each axis with variance 2 W and a mean equal to the centroid's corresponding coordinate. We used the clustered (CL) distribution with five clusters and variance 0.052 as the default setting for both the W and P datasets.

Algorithms. We compare the Jaccard and extended Jaccard similarity metrics using our user-centric similarity search algorithms (-similarity and m-NN). We have introduced a variant that takes advantage of previously processed RT OPk queries (which is still in use) to improve the bounds (in an in-memory R-tree), as defined. After the corresponding query is completed, the contents of this in-memory R-tree are discarded. The "-IMR" suffix in the labels of the corresponding graphs denotes experiments using the in-memory R-tree.

Metrics. The number of reverse top-k queries processed, the query execution time taken by each algorithm, and the similarity simUC,EU of the result to the answer obtained using the Euclidean distance are the key metrics we use. The Jaccard coefficient of the result sets (the number of common products in the answers divided by the size of their union) of the corresponding queries expresses this similarity. The algorithm of [4] was used for the mNN queries in Euclidean space. For the range queries, we ran a Euclidean NN query for the same product q that the -similarity query explores and retrieves as many items as the size of the result set of the similarity query. A value of simUC, EU far lower than one shows that the user-centric similarity question (m-NN/similarity) returns substantially different answers in our experimental evaluation.

Query: In all cases, we present average values based on 20 queries. The question points are chosen at random from a subset of P's data points. This subset includes either the nearest to axes origins products of P in the case of synthetic data sets, or products from the skyline set of P in the case of real data sets, to increase the likelihood of obtaining nonempty result sets.

Parameters: We run experiments with different dimensionalities d (2-6) and cardinality |P| (10K-100K), cardinality |W| (1K-15K), k (50-150), 0.7-0.95, and P for data distributions. The sample parameters are defined to the appropriate default values: k = 100, |W| = 1000, = 0.9, d = 4, and m = 10.

5. EXPERIMENTAL RESULTS

Uniform vs. Clustered set P. The performance of the similarity queries for the two separate data sets is shown in Figure 5.The in-memory R-tree version of the proposed algorithm for similarity queries reduces RT OPk evaluations dramatically (at least three times less), as shown in Figure 2a. As a result, the execution time is reduced (Figure 2b). Figure 2c shows a comparison of user-centric similarity search with conventional Euclidean distance similarity search and serves as a comparison of user-centric similarity search with conventional Euclidean distance similarity search. The graph clearly demonstrates that the retrieved results vary. For both cases of data distribution, Figure 4 repeats the experiment, this time using m-NN queries. The original algorithm and the one that uses the in-memory R-tree exhibit the same patterns as in the case of similarity queries.

Varying θ Figure 4 shows the sensitivity of similarity queries as the threshold value is changed. Higher similarity threshold values result in a small reduction in both the number of RT OPk executions and the execution time (Figure 4a) (Figure 4b). In Figure 4c, we see that higher threshold values increase the similarity of the two result sets when comparing the results sets of proposed queries and similarity queries in Euclidean space.

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 5, 2021, Pages. 3918 - 3927
Received 15 April 2021; Accepted 05 May 2021.

6. RELATED WORK

In the database and information retrieval cultures, top-k queries have been studied extensively. Based on available user preferences, such queries return the k most promising products [6], [8]. [9] deals with the problem of evaluating the accuracy of the top-k result set returned by an information retrieval system, such as when comparing search engine results. The authors discuss a variety of alternative measures and offer quick approximation algorithms for assessing some of them. Reverse top-k queries, on the other hand, are implemented in [4] and return the users who position a product (the query point) in their top-k result sets. Reverse top-k queries can be used to find influential products, with influence specified as the reverse top-k result set's cardinality [10]. Since it is directly related to the number of consumers who trust a specific product, this concept of influence is useful for market research. Despite recent techniques for evaluating reverse top-k queries, they are known to have substantial processing and I/O overhead, as a query usually needs the execution of several top-k queries to determine which customers prefer the queried product. RNN query returns all objects with the query object in their k nearest neighbors. The decision support role of finding the best location for a new store is the classic motivational example [12] of RNN. When faced with multiple location options, the approach is to choose the one that will draw the most customers. An RNN question returns customers who are more likely to prefer the new store over the current stores due to its geographic proximity. RNN queries can be implemented using a variety of methods [13], [14]. There are also algorithms that deal with data sets with higher dimensionality [15].The similarity introduced and used in RNN queries, where the responses are solely focused on object characteristics while users' expectations are ignored is not taken into consideration, our paper is somewhat different. Item-based collaborative filtering strategies [16] may have a similar intuition, but they suggest that customers have a taste of some products and thus rate them, while our methods suggest that customers get a taste of certain goods and rate them as a result Consider the case of a new product on the market or one that is still being designed during the production process.A collective filtering algorithm would be useless in both cases since there would be no ratings representing customers' views. Our system, on the other hand, does not require any prior knowledge of users' views about products because they express their expectations in a more general way, including a weighting factor for and product attribute, which is different from rating individual products.

7. CONCLUSION

We implemented a user-centric similarity paradigm in this paper, which considers user expectations when assessing product similarity. We showed through examples and experiments that user-centric similarity search can produce very different results than traditional metrics that focus solely on the goods and ignore the preferences that customers have expressed. Two interesting query types were defined, and efficient algorithms for their execution were proposed. We have also discussed about optimizations that help reduce execution times.

REFERENCES

[1] https://www.researchgate.net/publication/280101455_Literature_Review_of_Data_Mining_Applications_in_Academic_Libraries
[2] https://jpinfotech.org/user-centric-similarity-search/
[3] https://ieeexplore.ieee.org/document/7551120
[4] http://www.cs.umd.edu/~samir/498/topk.pdf
[5] https://ieeexplore.ieee.org/document/5447890
[6] http://www.ijirset.com/upload/2014/may/100_AStudyonReverse.pdf
[7] https://www.sciencedirect.com/science/article/abs/pii/S0022000016300459
[8] http://myweb.sabanciuniv.edu/rdehkharghani/files/2016/02/The-Morgan-Kaufmann-Series-in-Data-Management-Systems-Jiawei-Han-Micheline-Kamber-Jian-Pei-Data-Mining.-Concepts-and-Techniques-3rd-Edition-Morgan-Kaufmann-2011.pdf
[9] https://www.geeksforgeeks.org/measures-of-distance-in-data-mining/#:~:text=In%20a%20Data%20Mining%20sense,on%20the%20context%20and%20application.
[10] https://www.cs.princeton.edu/courses/archive/spr04/cos598B/bib/CharikarEstim.pdf
[11] https://www.ibm.com/docs/en/spm/7.0.9?topic=interfaces-user-preferences
[12] https://www.sciencedirect.com/topics/computer-science/euclidean-distance