Project Group W8-14

COMP90050 Advanced Database Systems

Winter Semester 2021

Survey on Top-K Queries

—

A Review of the Current Research on and Usage of Top-k

Prepared by:

Ghina Yashar (1274878)	gyashar@student.unimelb.edu.au
Yanze Mao (988142)	yanze.mao@student.unimelb.edu.au
Yiqing Ling (852955)	yiqingl1@student.unimelb.edu.au
Zihao Zhang (1151006)	zihaozhahng2@student.unimelb.edu.au

Abstract

Efficient and accurate processing of top-k queries is becoming more and more important in many interactive big data environments, such as distributed database systems and Web search. In this survey, we evaluate and classify different top-k processing techniques in different types of databases. Multiple aspects of current techniques are reviewed, including query models, data access methods, data and query certainty, ranking functions and more. We also look at new top-k variants and interpretations that focus on specific functions and use cases. In addition, we discuss the performance of top-k queries in different application domains.

1 Introduction to Top-K Queries

Top-k is a type of query that in its simplest traditional form:

1. Accepts as **input**:
    - Some set of data D (typically of objects in a tuple format)
    - A scoring function $S(d)$
    - A number k
2. **Calculates** a score $S(d)$ for $d \in D$
3. **Returns** the k objects $d_0 ... d_k$ that achieve the highest scores

Top-k queries are used in many areas and applications, including relational databases, XML document search, similarity search, uncertain data, and many more. They are also often adapted and used as the base for many more new query types that address specialised problem scenarios.

1.1 Common Approaches

In general, we classify top-k techniques into 1) indexing methods, and 2) threshold value filtering methods.

- The **indexing method** scores each instance with an index to build an index tree to access the top-k elements of each query to improve efficiency of the query response time with the help of additional storage space. Maintaining a sorted top-k index for the underlying objects based on the scoring function is very expensive.

- The **threshold value filtering method** finds the set of top k objects in the data set with a threshold value. It formulates a top-k query as a range selection query to limit the number of retrieved objects in the set. A range query is used to limit the retrieved set to the exact k necessary objects to answer the top-k query.

1.2 Top-k Research

The research on top-k queries is rich and varied. In this survey, we will look at a few different categories of top-k research papers, including:

- Optimisations of the traditional top-k query type
- Adaptations of top-k queries to make them more compatible with different data and database types
- Extensions of top-k queries to address different problem types
- The employment of top-k techniques in real solutions

2 Literature Review

2.1 Top-K Queries in General

2.1.1 Indexing method

The proposed method in “Answering Top-k Queries over a Mixture of Attractive and Repulsive Dimensions” (Ranu and Singh, 2011) is to access top-k by saving the data in a tree to make new queries to be fast, where the scoring function is based on the Manhattan distance between database objects and a given query point q. The solution by Ranu and Singh for top-k queries is to select the k nearest neighboring data points to the query points. Ultimately, we can project multi-dimensional data into a one-dimension index for comparison, which can be used as an index in constructing a balanced tree to store the data. Although $O(n)$ storage cost to construct the balanced tree is not cheap, such a process can be done before the query step. The query would be binary search costing $O(\log n)$ time, which ensures very short query time even if the data set is huge. However, the algorithm assumes we have a data set P of multidimensional points and limits the usage on data set with non-numeric attributes.

One example of a specialized indexing method is the onion T technique, introduced in “The Onion Technique: Indexing for Linear Optimization Queries” (Chang et al., 2000). The method is based on geometry, assuming that tuples are represented as n dimensional points. Chang et al. index objects with the convex hulls of these points as the boundary for elements in different layers. The geometric property of convex hulls ensures the top element is on the out-most convex hull layer and the second top element is on the out-most convex hull of the remaining elements. The strength of this onion T method is its efficiency in processing queries with small k problems. Even if the data set is huge, finding a few top elements with this algorithm is faster than traditional methods.

2.1.2 Threshold value filtering method

Computing immutable regions is a good example for threshold value filtering top-k query in “Computing Immutable Regions for Subspace Top-k Queries” (Mouratidis and Pang, 2012). In this paper, Mouratidis and Pang propose an algorithm to find the intersection of sets that are weighted differently. The algorithm first scans the set of k elements to be an immutable region, constrains it further and checks if there are any tuples entering or exiting the region. With the help of candidate pruning and thresholding, the set of tuples is checked with different dimensions. In the end, it determines the top k elements as well as the immutable region range for these elements to be the solution.

2.2 Uncertain Databases

In some query processing environments, finding exact query answers efficiently is overwhelming to the database. So, the accuracy of query answers is sacrificed to secure the performance efficiency. A top-k query can only report approximate answers from uncertain access of data. Alternatively, the uncertainty can be rooted in the nature of data. For example, the traffic speed measured by each sensor has a probabilistic distribution of each element’s existence on a continuous interval (Soliman et al., 2007). A query is not guaranteed to access the exact top k elements in the database. The interplay between ranking and uncertainty makes traditional techniques inapplicable since there is no deterministic guarantee of the existence of the proposed top k elements.

2.2.1 Collective method

In the paper "Efficient Processing of Top-k Queries in Uncertain Databases with x-Relations" (2008), Yi et al. find the set of k elements, which are most likely to be the query points in the data set, all together. First, the paper defines the solution to an uncertain query as the k tuples with the highest probabilities of being in the top-k solution. The process to find this involves repeatedly retrieving n tuples in order of their scores. This scan depth n is determined by k and the database size, and is the minimum number of scans required to ensure that the result is correct. The top k elements are found by comparing the probabilities of different scans and finding the largest. The selected k-element set in each probability comparison alternates either a single element or multiple elements.

The collective method of finding the top k elements all together can also be implemented on a stream of unknown data with a known distribution. In paper the "Sliding-window top-k queries on uncertain streams" (Jin et al., 2008), an uncertain data stream with a sliding window is processed with the goal of answering a probabilistic top-k-ranked query given a ranking function. To do this, a *compact set* is maintained during the scanning process by checking rank scores of new coming elements and leaving elements. The compact set remembers historical worst scores to ensure correctness and find the top k solution in high-speed data.

2.2.2 Marginal method

In "Top-k Query Processing in Uncertain Databases" (2007), Soliman et al. extract the set of k most probable elements marginally then iteratively find and pop out the best element k times. This method treats probabilities as the only scoring dimension, where a top-k query is a Boolean query that reports the k most probable query answers. In addition, it studies the interplay between the scoring and probability dimensions. Some scoring function is to order tuples, the probability of a k-length tuple vector T. The top–k elements are defined by comparing the summations of possible worlds probabilities where T is the top–k solution. The answer to an Uncertainty-k Rank query is a set of tuples that might not form together the most probable top–k vector. However, each tuple is a clear winner at its rank over all worlds. The algorithm materializes all possible worlds space, sorting each world individually, and merging identical top–k answers.

2.3 Dynamic Databases

Traditional top-k methods are extremely inefficiently when it comes to dynamic databases, in which objects change frequently. This is because they re-compute everything whenever a change happens. Improvements of top-k on highly dynamic circumstance are proposed in the following papers.

2.3.1 Hybrid Method

"Processing a Large Number of Continuous Preference Top-k Queries" (Yu et al., 2012) developed a fully dynamic solution to deal with such situations. This paper introduced a dynamic index which supports reverse top-k queries and proposed a query response surface (QRS) to intuitively represent the objects over the space of all possible preference vectors in a geometrical way. Each facet of QRS has a set of preferences sharing the same k-th ranked object. For the QRS-driven method, in addition to putting an index on objects, it maintains an index on each facet of QRS, and indexes of both would be updated if an object changes. This method successfully computes the set of affected preferences and the changes to the lists of top-k objects for each set of preferences. But this method can be too hard to compute especially for higher dimension dataset. For preference-driven method, it first finds the set of affected preferences if an object changes and runs a separate top-k query for each affected preference in the set, which can update indexes on objects efficiently but may be computationally costly if the set is large.

For retrieving the best from these two methods, the authors suggested combining them together where preference-driven is used for regions with few or no preferences and QRS for dense clusters of preferences. To further improve efficiency, they used approximation by indexing a small subset of objects and updating the list more efficiently. The results showed by using approximation, the computational time can be reduced significantly with a little cost of precision.

2.3.2 Top-k monitoring over sliding window

Mouratidis et al. (2006) proposed a different method to solve a similar problem. Their paper, "Continuous Monitoring of Top-k Queries over Sliding Windows", introduces two versions of sliding windows: the count-based Window containing the N most recent records, and the time-based Window containing all recent tuples that appeared within a fixed time. Only when the new tuples belong to a sliding window can they be considered as valid, and those valid tuples will be stored in the main memory and to indexed using a regular grid. When a top-k query is first run on the database, the computation module would compute the result by searching the minimum number of cells that have the target records. Then if the objects in the database change, top-k maintenance will only deal with the affected objects in these cells. For top-k maintenance, the paper describes two algorithms. One is the Top-K Monitoring Algorithm (TMA), which will re-compute the result of a top-k query when some objects of the current top-k answer become unavailable. The other is the Skyband Monitoring Algorithm (SMA), which reduces the size of the problem and frequency of computations by maintaining a subset of current results in the form of a k-skyband.

Extensive experimentation shows that both algorithms outperform the benchmark approach that uses the existing method for the same problem in terms of CPU time and space requirements, and SMA has even better performance than TMA for all parameter settings.

2.4 Graph Databases

Given the inherently complex graph structure of graph databases, running top-k query on them is a unique challenge. The following papers discuss methods that make it more feasible.

2.4.1 Top-k most representative graphs

In "Answering Top-k Representative Queries on Graph Databases" (2014), Ranu et al. developed an improved greedy algorithm to solve the problem of finding the k most representative objects in a graph database. Users need to identify the relevance function, size of the subset k and a distance threshold T. First, the relevance function is used to roughly categorize objects as relevant or not, then identify the k-most representative objects based on the set of possibly relevant objects identified previously, say L. One key criterion for measuring the degree of relevance of a set of graphs is the representation power π_s, which is defined as the proportion of relevant graphs. The paper proposes a baseline greedy algorithm which attempts to put the graph in L to the answer set when it can yield the biggest marginal gain in π_s. However, this algorithm requires to compute π_s for every pair of graphs in L and consume $O(n^2)$ computation time for each iteration, making it NP-hard and not scalable to large databases.

To make this algorithm computationally feasible, the authors introduced a new index structure named NB-index to index the T-neighbors of target graphs, which uses the combination of agglomerative clustering and Lipschite embedding. The improved algorithms would update π_s of a graph based on clusters and use embedding to speed up the process of indexing on required neighbors. Therefore, the algorithm can access the object more quickly, and since every graph is compared with one target point in each cluster, only linear time is required for computing π_s of all graphs.

2.4.2 Top-k for graph similarity

For finding the top-k most similar graphs from a graph database, Zhu et al. developed an efficient and accurate approach based on the maximum common subgraph (MCS) in their paper "Answering Top-k Graph Similarity Queries in Graph Databases" (2019). The graph distance is computed by the formula $Dist(q, g) = |E(q)| + |E(g)| - 2|E(mcs(q, g)|$, where q is the query graph, g is MCS, and *E(g)* represents the number of edges in graph g. It can capture structural connectivity more precisely, and the top-k graphs that have the smallest *Dist(q,g)* should be the result. However, computing MCS for every graph is NP-hard, so the authors introduced a lower bounds-based framework to prune unqualified graphs, which adopts the strategy that for each graph g in the given database, if the lower bound of *Dist(q,g)* is greater or equal to the greatest distance of the current top-k result, g should not be the top-k answer and can be pruned. The paper proposes two ways of computing the lower bounds: one is the Edge Frequency Based Approach, which computes the lower bound based on the frequency of distinct edges, and the other one is Adjacency List Based Approach, considering the weighted bipartite matching of adjacency list. By cutting off unqualified graphs during the query process, it could improve the efficiency dramatically without sacrificing the accuracy.

2.5 Top-K Query Variants

With the advancements of database technologies and the increasing activities and demands of information retrieval efficiency, there are strong needs to adapt the existing top-k query methods in order to make it useful in more situation and use cases. The properties of the top-k query variant papers to be discussed are as follows:

The relevant papers for Variant Top-K Queries			
Paper Title	Area	Variant Type	Field Position
Optimal Top-k Generation of Attribute Combinations Based on Ranked Lists (Lu et al., 2012)	Top-k,m	Top-K Variant	Foundation
Top-k Bounded Diversification (Fraternali et al., 2012)	Top-k diversity queries	Top-K Variant	Improvement
Reverse Top-k Queries (Vlachou et al., 2010)	Reverse top-k	Novel Top-K Interpretation	Foundation
Answering Why-Not Questions on Top-K Queries (He and Lo, 2012)	Why-Not Questions	Top-K Extension	Foundation
Answering Why-not Questions on Reverse Top-k Queries (Gao et al., 2015)	Why-not Questions	Reverse Top-K Extension	Foundation

2.5.1 Top-K,M

Top-k,m queries are designed specifically for problems where the following is given: 1) a set of groups, each containing a set of attributes, 2) each set of attributes is associated with a ranked list of tuples, and 3) tuples consist of an ID and a score, and are ranked in decreasing order of the scores. As a solution, top-k,m queries find the top-k combinations of attributes, where each combination includes one attribute from each group, according to the corresponding top-m tuples with matching IDs.

The paper "Optimal Top-k Generation of Attribute Combinations Based on Ranked Lists" (Lu et al., 2012) discusses a wide range of applications for top-k,m queries and illustrates this with a number of use-cases, including 1) the refinement of XML keyword queries by suggesting alternative terms that are guaranteed to have high-quality results in the database, 2) evidence combination mining in medical databases to predict or screen for diseases, and 3) in package recommendation systems, e.g. recommending trips, consisting of a combination of hotels, activities, and restaurants, based on the feedback of users who experienced each exact trip combination.

As the solution to a top-k,m problem consists of combinations of attributes in groups, whereas the solution to top-k problems consists only of tuples or objects, the top-k,m problem cannot be simply reduced to a top-k problem and requires a completely different approach to be solved.

A family of efficient algorithms are proposed in the paper, including a basic implementation named ULA and a more optimized version named ULA+.

2.5.2 Top-K Diversity Queries

Top-k diversity queries aim to find the best set of objects that are relevant to given user criteria and well distributed over a region of interest. Such queries can be extremely useful for exploring geo-referenced data. One example can be online map with spatial objects.

In the area of top-k diversity queries, the traditional method MMR (Maximum Marginal Relevance) evaluates the whole dataset to get such diversified top-k queries and can be costly and time consuming. Therefore, a more efficient MMR-correct algorithm PBMMR (Pull/Bound

Maximum Marginal Relevance) is illustrated in the paper “Top-k Bounded Diversification” (Fraternali et al., 2012). PBMMR requires following assumptions: 1. Objects are represented in a vector space and can be fetched through interfaces. 2. Objects grants sorted access either by relevance or by distance. PBMMR exploits spatial probing locations and the adaptive alternation of score-based and distance-based access to reduce the number of fetched objects. Moreover, the paper also introduce a PBMMR powered instance SPP(Space Partitioning and Probing), an instance using PBMMR with a pulling strategy of a tight upper bound. Experiments show that SPP can output the same quality of diversification as MMR does but with significantly less object access.

2.5.3 Reverse Top-K Queries

Reverse top-k queries are originally defined in 2010 in the paper “Reverse top-k queries” (Vlachou et al.). On the contrary to traditional top-k queries, which focus on user or customers’ perspective, the reverse top-k query type is introduced as a tool for manufacturers, which answers the question “which user preferences for a potential product result in such a product is in the top-k queries result set”.

In definition, top-k queries take a positive integer k and a user-defined weighting vector w and return a set of points satisfying several conditions. The reverse top-k queries oppose the top-k queries, which take a positive number k and a data points dataset S (if it is bichromatic instead of monochromatic version then a weighting vectors dataset W is also included) and return a collection of weighting vectors satisfying several conditions. The result collection is respectively a solution space for monochromatic version and a finite number of vectors for bichromatic version.

Along with the reverse top-k definition, monochromatic and bichromatic reverse top-k queries are two initially proposed versions. The former one is used when there is no prior knowledge of user preferences and the latter one is used when there is a data set that contains prior knowledge of user preference. In practice, monochromatic reverse top-k queries are used in order to estimate the impact of a product by return partitions of the solution space, thus the main utility is to get a geometric interpretation and help intuitively understand the problem. On the other hand, bichromatic reverse top-k queries can be more practical as this version can incorporate with user preferences database, returning key influential preferences that is essential for product development. Furthermore, a threshold-based algorithm (RTA) and an indexing structure (RTOP-Grid) are developed to increase the efficiency of bichromatic reverse top-k queries.

2.5.4 Why-Not Questions on Top-K Queries

“Why-not” questions raise when users question why their expected tuples do not show up in the query result. As an extension to this, “why-not questions on top-k queries” can have several ways to answer – the input integer k, the weighting vector or both. In addition, “why-not questions on dominating queries”, in which the dominating queries is a variant of top-k query which does not require weight set but uses dominated number of objects to set the ranking, is also a typical “why-not” question in top-k query field. The paper “Answering why-not questions on top-k queries” (He and Lo, 2012). address the why-not questions on top-k queries and dominating queries for the first time.

With a reverse top-k query answers “which weighting vector set make the object is in top-k result list” and a why-not question answers “why expected object is not in top-k result list”, a novel type of problem is defined by combining these two definitions – why-not questions on reverse top-k queries, which can be expressed in following sentence: “why expected weighting vector set is not in reverse top-k result list”. The semantics can be very useful to manufacturers as reflection analysis regarding the products or the customers in real life situation.

To answer such a why-not reverse top-k question in a decent manner, the paper “Answering why-not questions on reverse top-k queries” (Gao et al., 2015) prototypes a unified framework WQRTQ (Why-not Questions on Reverse Top-k Queries) which takes either a monochromatic reverse top-k query or a bichromatic reverse top-k query and corresponding why-not weighting vector set as inputs, and returns the refined reverse top-k query with minimum penalty.

2.6 Applications of Top-K Queries

2.6.1 Feature Matching Between Image Pairs

The paper “Feature Matching Based on Top K Rank Similarity” (Jiang et al., 2018) explores the use of top-k concepts in computer vision and pattern recognition tasks. Specifically, it proposes a method to match feature points between image pairs by comparing their local neighbourhood structures when represented as top-k ranking lists.

The method involves first searching the k nearest neighbours of a given feature point to obtain its ranking list, which denotes its local neighbourhood structure and the topology to be matched. Ranking lists of feature points in an image pair are then compared using an iterative optimization strategy that removes outliers and penalizes mismatches. Pairs that receive minimal penalties are labelled as matches.

Through the experimentation detailed in the paper, this method was able to generate better results on ten image pairs than some state-of-the-art feature matching algorithms.

2.6.2 Geographic Document Search

Geographic search engines are often inefficient due to the lack of an index that can simultaneously handle both the textual (i.e. query keywords) and spatial (i.e. query target location) aspects of geographic queries in an efficient manner. In the paper “IR-Tree: An Efficient Index for Geographic Document Search” (Li et al., 2011), such an index is proposed to work with a top-k document search algorithm to demonstrably outperform other state-of-the-art approaches.

The proposed document search algorithm discards of documents that are not textually and/or spatially relevant to the query as early as possible, postpones the calculation of the relevance of a documents until it is known to have a high chance to be in the top-k results, and terminates once the top-k documents with highest relevances are identified. These features significantly advantage the IR-Tree approach against other alternatives in terms of time efficiency and storage and computation requirements, as demonstrated in the experiments detailed in the paper.

2.6.3 Location Trace Similarity

Trace similarity search refers to identifying the location traces or trajectories that are most like a given query trace. Typical trajectory search services assume that the user trajectories to be searched and compared are stored on a centrally managed infrastructure prior to query execution. In contrast, the techniques proposed in the paper "Crowdsourced Trace Similarity with Smartphones" (Zeinalipour-Yazti et al., 2013) are decentralised and maintain the data in-situ (i.e., on the smartphone that generated the data), using instead a distributed top-K processing algorithm to answer trace similarity queries.

The proposed framework, named SmartTrace+, work as follows: 1) a query trace Q is posted by some centralized query processor QP, 2) each participating node (consisting of a smartphone) executes locally one or more inexpensive matching functions and returns the result to QP, 3) QP identifies some top candidate nodes, 4) each identified candidate node calculates and communicates back to QP its full similarity score, and finally, 5) QP identifies the top-k nodes whose trace best resembles Q.

The unique advantage of this framework is that similar traces can be identified without ever unveiling the target trajectories to the query processor. This protects user privacy and also eliminates the need to continuously transfer location data to the query processor, which can deplete the device battery and degrade the network health.

2.6.4 Similarity Search

In most cases, similarity computation is done in a manner that only considers items' attributes matrices distance. However, some cases require similarity for not only items' characteristics but also users' preferences. When those are taken into account, the similarity among products can be quite different (e.g., favourite lists). Thus, a recent paper "User Centric Similarity Search" (Deepa et al., 2021) exploits reverse top-k queries in the application of similarity estimation.

Reverse top-k queries are used in order to return such users' preferences as weighting vectors set while top-k queries are used to return the query point q for customers' returns. The similarity between products is achieved by same user preference satisfaction, which equals the similarity between their reverse top-k result set. The above framework is called user-centric similarity search. Jaccard coefficient can be an option to calculate the distance/similarity. However, it is way different from Euclidean space distance result. Therefore, a θ-similarity query and a m-NN query is defined and used to compute the similarity.

In addition, at implementation level, an extended Jaccard coefficient solution, which handles actual ranking position schema, and a dynamic tighter similarity bounds, which eliminates irrelevant points, are implemented for accuracy and efficiency consideration.

3 Comparison of Different Approaches & Papers

3.1 Performance of Different General Approaches

The indexing method designed by Ranu and Singh is more efficient in querying step than naive top k query approach without any index, especially when size of data is huge. Compared to classical indexing method by Ranu and Singh, the Onion T method proposed by Chang et al. is not efficient in finding top k elements with large k or data with high dimensions. The efficiency of onion T querying is lower than the classic method. However, it requires less storage, which leverages the storage and the query processing time. Moreover, Onion T algorithm performs very slowly in the worst case, but faster than theoretical speed on average. The main benefit of threshold value filtering method proposed by Mouratidis et al. is its adaptability in dynamic databases, the set of top k elements can be updated as high-speed data flow in and out, as it is faster to update the compact set of top k elements than updating a structured balance tree to store data. On the other hand, it is less efficient than indexing method in processing static data set. When k is large, finding the threshold value to suit k elements in filtered set is hard. This exacerbates the problem in efficiency of top k querying.

3.2 Comparison of Different Top-k Approaches on Dynamic Databases

For highly dynamic databases, general top-k algorithms fail to yield a satisfactory performance. This is because whenever an object in the given database is changed, the top-k result has to update by running another top-k query to ensure the result is up-to-the-second. Running continuous top-k queries using normal top-k algorithms that re-compute everything as long as a change happens wastes a large amount of time, memory space and computational energy. Even if it is feasible to run such top-k continuously, the result would suffer from serious time lag.

Both algorithms on this topic included in the survey consider a method to update results more efficiently by attempting to narrow down the computation range to the affected data or parameter and using other ways (e.g. index) to speed up the computation process.

For the approach proposed by Yu et al., one key contribution is it considers not only the change of objects in database, but also the change of preference used for identifying the top-k objects, especially useful when both of them change frequently, such as selecting top-k stock in an active market. The other method developed by Mouratidis et al., compared with the former method, it introduced two types of windows to filter out invalid new data which could save computation energy in the beginning. The top-k maintenance algorithm they used only deal with the affected and valid data, and re-compute the result if the current answer has changed, instead of running redundant queries.

3.3 Comparison of Different Top-k Approaches on Graph Databases

Two papers are selected on this topic, and they focus on different aspects. Ranu et al. aim to find the k most representative graphs, while the paper by Zhu et al. introduces a method to select the top k graphs that are most similar to a particular graph. Due to the complexity of graph structure, they both developed special and appropriate metrics to compare relationships between graphs according to the target problem, which are representation power and graph distance based on MCS respectively. Yet usually the computation for such metrics in graph databases is extremely heavy as every pair of graphs need to be compared in each iteration, therefore they developed different ways to make it computational feasible.

Ranu et al. proposed a new index structure that using agglomerative clustering and Lipschite embedding in combination, making the process of accessing objects faster but would require extra space to store index information. While Zhu et al. adopt a pruning method, that cuts off unqualified graphs during the query, based on the upper bound and lower bound they calculated according to the graph features and their goal. Compared with using an index structure, this pruning method can help save memory space but the computation for lower bound and upper bound would be more complex.

3.4 Motivation for Different Top-K Variants

As stated before, the simple top-k query type cannot satisfy users nowadays given the burst of demand in the data science field. Users need top-k queries to work in a manner that covers additional requirements and use cases. For the papers we have examined, there are different type of modifications on top-k queries. On one hand, there are variant top-k query types like top-k,m queries, which aim to find the optimal combination results, and diversity top-k queries, which find the optimal *k* results in the diversified manner. On the other hand, there are different interpretation and extensions for top-k queries, like reverse top-k queries, which take result points and return weighting vectors instead, and top-k relevant why-not questions, which refine the current top-k result list or reverse top-k weighting vector set list with user's specification.

The discussed top-k variant papers have the following usage comparisons:

The relevant papers for Variant Top-K Queries		
Paper Title	Top-k Variant and Use Case	Example
Optimal Top-k Generation of Attribute Combinations Based on Ranked Lists (Lu et al., 2012)	Top-k,m - Find the optimal combination tuples of top-k results.	As a basketball manager, aims to find best combinations of two players that outplays any other.
Top-k Bounded Diversification (Fraternali et al., 2012)	Diversity top-k - Find top-k results in a diversification manner.	As a tenant, aims to find optimal houses in a range of area consists of several locations.
Reverse Top-k Queries (Vlachou et al., 2010)	Reverse top-k - Take a points list and return weighting vectors set instead.	As a manufacturer, wants to get or pair the user preference for potential products.
Answering Why-Not Questions on Top-K Queries (He and Lo, 2012)	Why-not Q on top-k - Refine the top-k query to include the expected missing items.	As a user, wants to know the reason that the expected item(s) is absent from result and refine the top-k query.
Answering Why-not Questions on Reverse Top-k Queries (Gao et al., 2015)	Why-not Q on reverse top-k - Refine the query to include expected missing weighting vector set.	As a user, wants to know the reason that the expected weighting vector set(s) is absent from result and refine the reverse top-k query.

4 Conclusion and Outlook

4.1 Future Directions

Given the current state of top-k queries and all the papers we have examined, we predict that future developments will focus on efficiency, accuracy and adaptability.

- **Efficiency:** For indexing method, a more focused strategy can be developed in the future to define a more efficient mapping function which would lead to more effective pruning of the search space. Onion T is one of good example of scoring function that efficiently. While for threshold value filtering method, the speed of finding corresponding threshold value to filter k elements can be increased with a well designed algorithm. Algorithms with better efficiency are worth developing, especially in process top-k queries with large k values.
- **Accuracy:** In the future, uncertain models can be developed further for finding the set of top k elements more accurately. It is ideal to obtain the set of k elements with less theoretical variance inherited from variance of uncertain data.
- **Data Adaptability:** In the future, proper solutions to cope with more complex uncertain models can be developed. For example, algorithms on correlated uncertain database, where each tuple appears with a probability dependent of other tuples, need developing.
- **Use Case Adaptability:** One possible future direction is that more top-k variants, extensions, interpretations, and combinations of them, will come out to further cover the user demands. With the call of demand draws more attention, a new variant that is typically designed for that demand should be created very quickly as the demand in nature is the idea and guidance for development. In addition, an extension is rather simple as well. For example, the why-not questions can be posed on different variants and interpretations in order to improve specific variant outcome. However, a novel interpretation is less likely to occur comparing to the other variant types given the top-k queries consists of only k, weighting vectors and result points, and once the new interpretation comes up, it will surely bring a burst in the development and possibilities in top-k queries, just like reverse top-k queries did.

4.1.1 One Potential Application Proposal

A novel top-k implementation can be a two-way self-adaption top-k query, which takes k, an initial user weighting vector set and an initial data points set. Using the two initial sets to train the association between its top-k query result points and its reverse top-k query result weighting vectors, and using why-not question to ensure a minimum-change basis on the combination of two initial sets. The query can then evolve in a dynamic database by applying above steps and return our required top-k or reverse top-k queries result in a continuous and reflective manner.

4.2 Conclusion

We have surveyed different top-k processing techniques in different types of databases. From the papers that we collected and analysed, research in the field of top-k queries can be divided into several subdivisions, including the general top-k algorithmic improvements, different data type adaptations, different variants and interpretations, and applications. We discussed the details of different algorithms to illustrate how they cope with the different challenges. Furthermore, we compare and contrast the papers in order to achieve a better view about the performance and the differentiating factors between them. Finally, some expectations for the future of top-k queries are included based on our analysis of the current work.

References

Chang, Y.-C., Bergman, L., Castelli, V., Li, C.-S., Lo, M.-L., & Smith, J. R. (2000). The onion technique: Indexing for linear optimization queries. *SIGMOD Rec.*, *29*(2), 391–402. https://doi.org/10.1145/335191.335433

Deepa, K., Anjali, J., Devika, R., & Dharshana, S. (2021). User centric similarity search. *Annals of the Romanian Society for Cell Biology*, 3918–3927.

Fraternali, P., Martinenghi, D., & Tagliasacchi, M. (2012). Top-k bounded diversification, 421–432. https://doi.org/10.1145/2213836.2213884

Gao, Y., Liu, Q., Chen, G., Zheng, B., & Zhou, L. (2015). Answering why-not questions on reverse top-k queries.

He, Z., & Lo, E. (2012). Answering why-not questions on top-k queries. *IEEE Transactions on Knowledge and Data Engineering*, *26*(6), 1300–1315.

Jiang, J., Ma, Q., Lu, T., Wang, Z., & Ma, J. (2018). Feature matching based on top k rank similarity. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2316–2320. https://doi.org/10.1109/ICASSP.2018.8461504

Jin, C., Yi, K., Chen, L., Yu, J., & Lin, X. (2008). Sliding-window top-k queries on uncertain streams. *PVLDB*, *1*, 301–312. https://doi.org/10.1007/s00778-009-0171-0

Li, Z., Lee, K. C., Zheng, B., Lee, W.-C., Lee, D., & Wang, X. (2011). Ir-tree: An efficient index for geographic document search. *IEEE Transactions on Knowledge and Data Engineering*, *23*(4), 585–599. https://doi.org/10.1109/TKDE.2010.149

Lu, J., Senellart, P., Lin, C., Du, X., Wang, S., & Chen, X. (2012). Optimal top-k generation of attribute combinations based on ranked lists. *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, 409–420. https://doi.org/10.1145/2213836.2213883

Mouratidis, K., Bakiras, S., & Papadias, D. (2006). Continuous monitoring of top-k queries over sliding windows. *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, 635–646. https://doi.org/10.1145/1142473.1142544

Mouratidis, K., & Pang, H. (2012). Computing immutable regions for subspace top-k queries. *Proc. VLDB Endow.*, *6*(2), 73–84. https://doi.org/10.14778/2535568.2448941

Ranu, S., Hoang, M., & Singh, A. (2014). Answering top-k representative queries on graph databases. *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, 1163–1174. https://doi.org/10.1145/2588555.2610524

Ranu, S., & Singh, A. K. (2011). Answering top-k queries over a mixture of attractive and repulsive dimensions. *Proc. VLDB Endow.*, *5*(3), 169–180. https://doi.org/10.14778/2078331.2078333

Soliman, M. A., Ilyas, I. F., & Chen-Chuan Chang, K. (2007). Top-k query processing in uncertain databases. *2007 IEEE 23rd International Conference on Data Engineering*, 896–905. https://doi.org/10.1109/ICDE.2007.367935

Vlachou, A., Doulkeridis, C., Kotidis, Y., & Nørvåg, K. (2010). Reverse top-k queries. *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*, 365–376.

Yi, K., Li, F., Kollios, G., & Srivastava, D. (2008). Efficient processing of top-k queries in uncertain databases with x-relations. *IEEE Transactions on Knowledge and Data Engineering*, *20*(12), 1669–1682. https://doi.org/10.1109/TKDE.2008.90

Yu, A., Agarwal, P. K., & Yang, J. (2012). Processing a large number of continuous preference top-k queries. *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, 397–408. https://doi.org/10.1145/2213836.2213882

Zeinalipour-Yazti, D., Laoudias, C., Costa, C., Vlachos, M., Andreou, M. I., & Gunopulos, D. (2013). Crowdsourced trace similarity with smartphones. *IEEE Transactions on Knowledge and Data Engineering*, *25*(6), 1240–1253. https://doi.org/10.1109/TKDE.2012.55

Zhu, Y., Qin, L., Yu, J., & Cheng, H. (2019). Answering top-k graph similarity queries in graph databases. *IEEE Transactions on Knowledge and Data Engineering*, *32*, 1–1. https://doi.org/10.1109/TKDE.2019.2906608