Probabilistic Top-\(k\) Query Processing Using Central Limit Theorem

GRADUATE PROJECT REPORT
Submitted to the Faculty of
Department of Computing Sciences
Texas A&M University-Corpus Christi
Corpus Christi, TX

in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science
By

Sharath Ranga
Summer 2011

Committee Members

Dr. Long-zhuang Li  
Committee Chairperson

Dr. Mario Garcia  
Committee Member

Dr. John Fernandez  
Committee Member
ABSTRACT

Top-k query processing is a widespread field of research. Its application can be used in many fields like wireless sensor networks, mobile ad-hoc networks, peer-to-peer networks and many more. The basic problem in top-k query processing is that, a single algorithm cannot be used as a solution to the problem of top-k query processing because there are many types of top-k query processing. The algorithm has to be based on the situation, the classification and the type of database and query model. The research method used in this project provides a solution to the problem of generating the probable top-k query results. The algorithm also provides the probability that the results are more likely in the top-k result set. The most prominent algorithms in the field of top-k query processing techniques are FA (Fagin Algorithm), TA (Threshold Algorithm). The algorithm used in this project is built upon the Threshold Algorithm making it applicable for a wider range of top-k query processing problems with better efficiency. In this project the algorithm is implemented on sorted data. The confidence for the results is also assigned. The confidence is calculated by using the cumulative frequency distribution functions and Central Limit Theorem. Histograms implemented in the Threshold Algorithm for the data approximation are replaced with the graphs based on Central Limit Theorem to approximate the probability of the results from the data aggregated. Implementation of the Threshold Algorithm with the concepts of Central Limit Theorem result in fast and efficient top-k query processing.
# TABLE OF CONTENTS

Abstract ........................................................................................................................................... ii

Table of Contents ........................................................................................................................... iii

List of figures .................................................................................................................................. v

1. Background and Rationale ........................................................................................................ 1
   1.1 Top-k Query Processing Techniques .................................................................................... 1
      1.1.1 Query Model ..................................................................................................................... 3
      1.1.2 Data and Query Uncertainty ........................................................................................... 4
      1.1.3 Data Access ...................................................................................................................... 4
      1.1.4 Implementation Level ....................................................................................................... 5
      1.1.5 Ranking Functions ........................................................................................................... 6
   1.2 Fagin Algorithm ....................................................................................................................... 7
   1.3 Threshold Algorithm ............................................................................................................... 9

2. Probabilistic Top-k query Processing using Central Limit theorem ...................................... 11
   2.1 Probabilistic Top-k Query Processing Techniques ............................................................... 11
   2.2 Pros and Cons of Threshold Algorithm and Histograms .................................................... 13
   2.3 The Central Limit Theorem Concept .................................................................................... 14

3. System Design .......................................................................................................................... 16
   3.1 Threshold Algorithm with Central Limit Theorem (TA-CLT) ............................................ 16
   3.2 Calculating Confidence ......................................................................................................... 18

4. Evaluation and Testing ............................................................................................................. 21
   4.1 Testing with various values of $k$ ......................................................................................... 21
4.2 Testing with the range of data items in the lists ........................................ 25
4.3 Evaluation ........................................................................................................ 28
5. Future Work ........................................................................................................ 35
6. Conclusion ........................................................................................................... 36
Bibliography and References .................................................................................. 37
LIST OF FIGURES

1. Figure 1: Hierarchy of Top-k query processing.................................................2
2. Figure 2: Positions of the data items belonging to three lists in a database with their Local scores.................................................................8
3. Figure 3: Positions of the data items belonging to three lists in a database with their Local scores and overall scores........................................9
4. Figure 4: A Framework for top-k processing in uncertain databases................12
5. Figure 5: Implementation of the stopping mechanism in TA-CLT ...................17
6. Figure 6: Time in milliseconds vs. different values of k for a database with 100 rows...22
7. Figure 7: Time in milliseconds vs. different values of k for a database with 1000 rows.................................................................................22
8. Figure 8: Time in milliseconds vs. different values of k for a database with 5,000 rows.................................................................................23
9. Figure 9: Time in milliseconds vs. different values of k for a database with 10,000 rows.................................................................................23
10. Figure 10: Time in milliseconds vs. different values of k for a database with 50,000 rows.............................................................................24
11. Figure 11: Time in milliseconds vs. different values of k for a database with 100,000 rows.............................................................................24
12. Figure 12: Time in milliseconds vs. number of data rows in a database for k=3........25
13. Figure 13: Time in milliseconds vs. number of data rows in a database for k=5........26
14. Figure 14: Time in milliseconds vs. number of data rows in a database for k=10......26
15. Figure 15: Time in milliseconds vs. number of data rows in a database for k=20......27
16. Figure 16: Time in milliseconds vs. number of data rows in a database for k=30......27
17. Figure 17: Time in milliseconds vs. number of data rows in a database for $k=50$.

18. Figure 18: A graph showing the comparison between the tuples retrieved by both algorithms from a database with 100 rows and 3 attributes for various values of $k$.

19. Figure 19: A graph showing the comparison between the tuples retrieved by both algorithms from a database with 100 rows and 3 attributes for various values of $k$.

20. Figure 20: A graph showing the comparison between the tuples retrieved by both algorithms from a database with 100 rows and 3 attributes for various values of $k$.

21. Figure 21: A graph showing the comparison between the tuples retrieved by both algorithms from a database with 100 rows and 3 attributes for various values of $k$.

22. Figure 22: A graph showing the comparison between the tuples retrieved by both algorithms from a database with 100 rows and 3 attributes for various values of $k$.

23. Figure 23: A graph showing the comparison between the tuples retrieved by both algorithms from a database with 100 rows and 3 attributes for various values of $k$.

24. Figure 24: A cumulative frequency table for the data elements with the Mean Threshold Values.

25. Figure 25: Number of tuples visited by various confidence percentages of TA-CLT.
1. BACKGROUND AND RATIONALE

Top-k query results are more critical in the current data integration and data centric world. The internet is filled up with a huge amount of data and there is a need for good top-k query processing techniques [Dedzoe, 2010] [Bhide, 2009] [Sasaki, 2010] [Chuanwen Li, 2010]. The emerging field of top-k query processing needs various mechanisms to generate query results faster and more efficient. A top-k processing technique used must be adaptable to various working environments. A typical top-k processing technique generates top-k results from huge databases. Databases with multiple lists or tables with huge number of tuples and rows are joined to form a huge database. The taxonomy of top-k query processing techniques and the famous Fagin and Threshold Algorithms are explained briefly in this section.

The Literature Review section is about the background of the top-k query processing. The Probabilistic top-k query processing using Central Limit Theorem section describes the basic working of the Probabilistic top-k query processing technique, pros and cons of the current version of the Threshold Algorithm and the usage of the Central Limit Theorem. The other following sections are self-explanatory.

1.1 Top-k Query Processing Techniques

Top-k query processing techniques are used for optimizing query executions. In order to find a small and specific number of the top results or results with aggregated values, top-k query
processing techniques are used. Top-\(k\) query processing has a huge taxonomy and hence top-\(k\) query processing techniques have vast applications in fields like sensor networks, peer-to-peer networks, distributed networks and data engineering [Liu, 2010] [Deutch, 2009] [Chen, 2010]. It is classified into various types based on the Query Model, Data Access Methods, Implementation Level, Data and Query Certainty and Ranking Function used by the different databases in various systems. Figure 1 below, shows the detailed classification of the top-\(k\) query processing where each aspect of a regular top-\(k\) query processing method is further divided into different sub-categories[Ilyas,2008].

![Figure 1: Hierarchy of Top-\(k\) query processing [Ilyas, 2008]](image)

All the top-\(k\) query processing techniques available belong to one of these classification nodes, which can be described by the path from the parent node to the leaf node. From Figure 1, it is clear that the hierarchy of the top-\(k\) query processing is a complex and vast field of research. There are many levels and different types at each node in the classification tree. As a system has
databases with different types, accordingly there are different top-k query processing techniques for each different database system. In this project, the algorithm is implemented on sorted data and the ranking function used is monotonic. Implementation is at application level. A brief explanation of various types of top-k query processing techniques is as follows [Aggarwal, 2009] [Ilyas, 2008]:

1.1.1 Query Model

Top-k query processing techniques use various query models to maintain the scores of the data objects. The various and most popular query models as are

a. Top-k Selection – In a top-k selection query model, the base tuples have scores attached to them. As a result, k tuples with highest scores are given as output by a typical top-k selection query. The scores attached to the tuples are dependent on the scoring functions. The scoring functions have to consider the overall scores of various tuple attributes.

b. Top-k Aggregate – In a top-k aggregate model the tuple groups have scores attached to them. Here in the model the scores are not attached to individual tuples. As a result, k tuple groups with highest scores are given as output. Group aggregate functions such as sum, average, mean are used for computing the group scores.

c. Top-k Join – In a top-k join query model, the join results of the tuples have the scores attached to them unlike to the base tuples as in top-k selection query model. As a result, k join results with highest scores are given as output by this model. The join is
based on some random join functions and the scores to the join results are based on some scoring functions.

1.1.2 Data and Query Uncertainty

Few query processing environments such as sensor networks, tracking the weather from the remote nodes and tracking moving objects involves probabilistic or uncertain data. In such processing environments, the top-\(k\) query answers are uncertain in nature. Hence the data for top-\(k\) processing is classified based on certainty. Data and query certainty classification of the top-\(k\) query processing techniques is as follows [Aggarwal, 2009]:

a. Exact Methods over Certain Data – All the top-\(k\) query processing techniques come under this classification if those use the deterministic data to process the deterministic top-\(k\) queries.

b. Approximate methods over Certain Data – All the top-\(k\) query processing techniques come under this classification if those use the deterministic data to process. But in these types of techniques, approximate results are generated compromising accuracy over performance. These approximate answers are mapped with probabilities to show the closeness to the exact results.

c. Uncertain Data – All the top-\(k\) query processing techniques come under this classification if those deal with the processing of probabilistic data.

1.1.3 Data Access

This classification dimension of the top-\(k\) query processing techniques deals with the access type assumed by the processing technique. The major types of data access available are sorted and
random access. The way which the query processing technique gains access to the database elements has a great impact on the design of the underlying top-k processing technique. The effect of the data access on a top-k processing technique is well explained in sections 1.2 and 1.3. The classifications of the top-k processing techniques based on the data access on the databases are as follows [Ilyas, 2008]:

a. Both Sorted and Random Access - Top-k query processing techniques which belong to this category believe the presence of both the access types on all the available data sources in the environment. Threshold algorithm belongs to this category.

b. No Random Access – Top-k query processing techniques which belong to this category believe the presence of only sorted access on all the available data sources in the working environment. This sorted access depends on the scores of the data objects in the data sources.

c. Sorted Access with Controlled Random Probes – Top-k query processing techniques which belong to this category believe the presence of at least one sorted access data source. The random access is limited only to reveal the composite score of the top-k results.

1.1.4 Implementation Level

Top-k query processing techniques are also classified based on the integration of the technique with the database system. The top-k processing technique can be integrated at higher level on the top of the database engine. Through this approach the top-k processing technique can be easily
extended as they are separated from the underlying query engine. In this approach, the top-\(k\) technique can be added without disturbing or making any changes to the query engine.

The other way to embed the top-\(k\) processing technique and the database system is by making modifications to the principle query engine. By these modifications the engine can understand the structure of the top-\(k\) processing technique and its scoring needs for the top-\(k\) results during the basic levels of the query planning and execution. Such approaches have greater impact on various factors like query efficiency, optimization and query processing. There are two levels at which most of the top-\(k\) processing techniques can be embedded with the database systems [Aggarwal, 2009]:

a. Application Level - As mentioned above, all of the top-\(k\) query processing techniques those work at the outer layer of the database engine belongs to this category. However there are some special cases with some techniques which take support of the specialized top-\(k\) index or the materialized views.

b. Query Engine Level – All the top-\(k\) techniques embedded with database engine at the query engine level have to make modifications to the core query engine so as to integrate with the database system. The query engine is now available for rank based processing and optimization as it is modified and knows the ranking requirements of the top-\(k\) processing technique.

1.1.5 Ranking Functions

The most important classification of the top-\(k\) query processing techniques is the ranking function. The ranking function has a greater impact on the design of the top-\(k\) query processing
technique. The classification of the top-$k$ techniques is based on the limitations they enforce on the ranking function used.

a. Monotone Ranking Function - This is the most common and efficiently used ranking function. Monotone ranking function suits many of the top-$k$ processing techniques for various database systems. Monotone ranking function has prominent properties which yield in efficient top-$k$ processing.

b. Generic Ranking Function – This ranking function is mainly used in specific top-$k$ query processing for constrained function optimization.

c. No Ranking Function – This ranking function is used by the techniques which deal with skyline related queries.

“Skyline query is a query that returns the objects that are not dominated by any other objects that are restricted to a set of dimensions”. A deep understanding of skyline query is out of the scope of the project.

1.2 Fagin Algorithm

Fagin Algorithm and Threshold Algorithm are the most efficient and well known algorithms for finding the top-$k$ results [Fagin, 2001]. Threshold Algorithm itself is based on Fagin Algorithm. The algorithms are illustrated with an example in Figure 2. The example has three lists with respective local scores for the ten data elements in each list. The data elements are represented with a notation $d_i$. Both algorithms are supposed to find top-3 results, i.e. $k = 3$. 
Fagin and Threshold Algorithms have stopping functions to stop the sorted access and return the top-
$k$ results. Fagin Algorithm stops the sorted access mechanism only after seeing at least $k$ data
items in all the three lists. Here in Figure 2, to get the top $k$ results the algorithm has to look into
databases where it finds at least 3 data elements in all the three local tables. Here it is at position
8 where the data items $d_1, d_3, d_5, d_6,$ and $d_8$ are found in all the three lists. Not until position 8
there were 3 elements in all the local sources. At position 7, there were only 2 data items $d_5$ and
$d_8$ that are found in all the three lists. For the remaining data elements $d_2$ and $d_4$, which were not
in all the three lists, the algorithm gets the random access from the respective lists for the local
scores. Fagin Algorithm then calculates the overall scores of the data items and returns the top

---

**Figure 2:** Positions of the data items belonging to three lists in a database with their local scores.

<table>
<thead>
<tr>
<th>Position</th>
<th>List 1</th>
<th>List 2</th>
<th>List 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Data item</td>
<td>Local score $s_1$</td>
<td>Data item</td>
</tr>
<tr>
<td>1</td>
<td>$d_1$</td>
<td>30</td>
<td>$d_2$</td>
</tr>
<tr>
<td>2</td>
<td>$d_4$</td>
<td>28</td>
<td>$d_6$</td>
</tr>
<tr>
<td>3</td>
<td>$d_9$</td>
<td>27</td>
<td>$d_7$</td>
</tr>
<tr>
<td>4</td>
<td>$d_3$</td>
<td>26</td>
<td>$d_5$</td>
</tr>
<tr>
<td>5</td>
<td>$d_7$</td>
<td>25</td>
<td>$d_9$</td>
</tr>
<tr>
<td>6</td>
<td>$d_8$</td>
<td>23</td>
<td>$d_1$</td>
</tr>
<tr>
<td>7</td>
<td>$d_5$</td>
<td>17</td>
<td>$d_8$</td>
</tr>
<tr>
<td>8</td>
<td>$d_6$</td>
<td>14</td>
<td>$d_3$</td>
</tr>
<tr>
<td>9</td>
<td>$d_7$</td>
<td>13</td>
<td>$d_{14}$</td>
</tr>
<tr>
<td>10</td>
<td>$d_{11}$</td>
<td>10</td>
<td>$d_{14}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
three data items with the highest scores computed so far from all the data elements which were visited. In this case the top $k$ results are $[d_3, d_5, d_8]$.

### 1.3 Threshold Algorithm

The usage of top-$k$ query processing in Threshold Algorithms can be witnessed in Figure 3. Figure 3 illustrate the implementation of the top-$k$ query processing in Threshold Algorithm [Fagin, 2001]. The Figure 3 below is an example of a database with three rows with data elements and their local scores. In order to keep it simple, the example is explained with a few number of data elements. The working of the Threshold algorithm is explained here with the same example illustrated above in Figure 2 which is used to explain Fagin Algorithm. The efficiency and performance can be compared with the data position at which the random access is started earlier in Threshold than in Fagin Algorithm. In reality, the databases are huge and the algorithms have to go through millions of data elements to come up with the top-$k$ query results. In Threshold Algorithm, the overall scores are aggregated. The top-$k$ results are given within the position where the Threshold Value is less than or equal to any of the overall scores of the data elements in the result buffer. In this case it is position 6 as marked in the Figure 3 [Arai, 2007] [Fagin, 2001] [Das, 2004]. The scores of the data elements ($d_1d_3d_3$) in the first row are considered as the top-3 results first. The highest of these 3 elements is compared with the Threshold Value at the first position.
As it is (88>70) less, the algorithm then considers the next three data elements however only the 3($d_3d_4d_5$) highest of these 6($d_1d_2d_3d_4d_5d_6$) data elements are maintained in the top-3 buffer. The highest score of these 3 data elements is now compared with the Threshold Value at that position (84>70). The process is repeated until the top-3 buffer has scores greater than the Threshold Value at that position. Here it is at position 6 where the top-3 buffer has the values $d_3d_5d_8$ and the value of $d_8$ is greater than the Threshold value at that position (71>63).

These algorithms can only generate the top-$k$ results, but they cannot generate the probability for the results. This brings the probabilistic top-$k$ query processing into the picture. According to [Ilyas, 2008], probabilistic top-$k$ query processing is the most challenging and futuristic mechanism.
2. PROBABILISTIC TOP-k QUERY PROCESSING USING CENTRAL LIMIT THEOREM

This project includes two important contributions to the field of the top-k query processing.

a. With the implementation of Central Limit Theorem (CLT) on Threshold Algorithm, the efficiency and approximation of the probabilities is improved. The top-k results are given by processing much less tuples as compared to the original Threshold Algorithm. This is a major contribution to the field of Probabilistic top-k query processing techniques.

b. With proper implementation, the usage of CLT can also be extended to various types of top-k query processing techniques as mentioned in the Figure 1, the taxonomy of top-k query processing techniques, for generating faster and efficient top-k results.

2.1 Probabilistic Top-k Query Processing Techniques

These types of processing techniques basically deal with uncertain data so as to ensure some probabilities along with the scores generated by the ranking functions. These probabilities along with the score ensure confidence in the results [Re, 2007]. A framework proposed by [Solman, 2007] for top-k query processing uncertain databases is shown Figure 4.
A typical uncertain top-k processing has two layers – Tuple access layer and Processing Layer – which are discussed below:

Tuple Access Layer: As shown in the figure this layer is responsible for the basic functionalities like ranking, score based indexing and traditional query processing. In order for the processing layer to gain access to the uncertain tuples, the uncertain data and probabilistic dependencies are stored here in the database [Solman, 2007].

Processing Layer: based on the rules of the query rule engine, upon getting the access to the uncertain tuples, the top-k processing is done to generate the most probable answers.
Threshold Algorithm does not have the probabilistic features in order to behave like a probabilistic top-
k processing technique. Hence the main idea is to provide the Threshold Algorithm with the probabilistic feature with the help of Central Limit Theorem and probability density functions.

An anytime measure top-
k algorithm is introduced with the concepts of probability density functions along with Threshold Algorithm to reduce the running cost of the top-
k processing techniques [Arai, 2007]. According to [Arai, 2007] an anytime algorithm generates the top-
k results at any point of the execution with a probability factor. The probability factor is called confidence to ensure the presence of the results in top-
k results set. In an anytime algorithm the accuracy of the results increases as the time increases. The confidence is generated by using the concepts of Histograms and probability density functions (pdf). Histograms consume a huge amount of time and take a lot of processing power to generate the top-
k results. The following section lists the pros and cons of Histograms.

2.2 Pros and Cons of Histograms and Threshold Algorithm

The basic relation between probability distributions and histograms is that they are used to evaluate the probability distribution of a given variable by illustrating the frequencies of the readings varying in particular range of value. The area under the histograms of the probability density function always adds up to 1.

Histograms are used in the Threshold algorithm for approximation of the PDF (Probability Density Functions). Histograms do not impose any restrictions nor need any
assumptions about the distributions that are approximated in the Threshold Algorithm. Histograms are used in Threshold Algorithm to generate the probabilities of the tuples. Histograms are based on the bucket size and the convolution which is a costly process [Arai, 2007].

2.3 The Central Limit Theorem Concept

The algorithm involves the joint collected Threshold Values from all the tables; Central Limit Theorem is applied on the Threshold values of the tuples. According to Central Limit Theorem, the means of an arbitrary finite distribution are always distributed according to normal distribution. Similarly in histograms, the distribution represented by the Central Limit Theorem is normal, and the area under the curve is added to 1. The Central Limit Theorem applied on the probability distribution does not require any buckets and hence the probability and confidence acquired are unique and independent of value $k$, where $k$ is the number of top results.

The list for the Threshold values of the tuples in databases is calculated and the means of these Threshold values are calculated and referred as Mean Threshold Values. These means of the values are considered to find the top-$k$ results by comparing them with the overall scores of the data elements as explained in the section 1.3.

The reason CLT is chosen for this project is because the data involved in finding top-$k$ results is huge and distributed randomly. Applying CLT on these random Threshold Values in the databases yields a normal distribution. Whenever CLT is applied to a series the shape of the
distribution curve is bell shaped. As the distribution is normal, it is now easy to find the probability density function. The working of the probability density function is explained in the later sections.
3. SYSTEM DESIGN

The system is basically developed using Threshold Algorithm mentioned in [Arai, 2007] [Fagin, 2001] and the concepts of Central Limit Theorem with frequency distribution. The major contribution of the algorithm is to provide the user with top-\(k\) results and probability attached to the results. The whole algorithm is implemented in c#. The actual algorithm implementation deals with the huge set-up of databases which is hard to set up. As the project deals with a new algorithm the algorithm is tested with a wide range of data. The testing of the algorithm is done with an excel file with huge rows of data. However the scalability of the algorithm depends on the environment it is used. The algorithm is tested with an excel file with rows ranging from 100 to 100,000. The program take in the excel file copies the data onto the console and the algorithm is implemented on this data. The testing of the algorithm is explained in section 4.

3.1 Threshold Algorithm with Central Limit Theorem (TA-CLT)

According to the Threshold Algorithm the scores of the data items in the rows are added together to be called Threshold Value. In this algorithm, in order to implement the Central Limit Theorem the means of the values of the data items at a particular position are calculated and referred as Mean Threshold Values. So now the new row with threshold values, which is the sum of the three values in the original Threshold Algorithm, is replaced with the means of the corresponding data items at each respective data position. A frequency distribution graph of the means is plotted to form what is called as the probability density functions. However the
stopping mechanism will be the same as that of the original Threshold Algorithm. The stopping function for the random access of the local scores can be explained in Figure-6 below. The same example is used to illustrate the stopping mechanism as mentioned in the sections 1.1 and 1.2.

From the above diagram it is clear that at the same position 6, the random access for the local scores has taken place. The Mean Threshold Value is the mean of the local scores at a particular position and the overall score in TA-CLT is the mean of the scores of each particular data elements. The top-k results are generated as similar to TA but the difference is that the number of tuples visited to generate the top-k results is less, making TA-CLT faster. In TA-CLT the number of tuples to be visited depends on the distribution, mean, standard deviation and confidence.

The top-k results are generated by assigning scores to the data elements, the data element with highest score is sent to the result set of size k. The elements with next highest scores are compared with the Mean Threshold Value and if the Mean Threshold value is greater, then the
next data element is considered or else it is considered as the highest scored element and is added to the result set. This procedure continues until the result set has $k$ data elements with highest scores. Whenever an element is added to the result set it is compared with the rest of the elements in the result set before discarding from the set. However the number of data elements to be visited depends on the confidence percentage the user wants to see.

The algorithm is implemented in C# and executed with Microsoft Visual Studio 2010. The input file must be placed in the C: directory. The input file is a typical excel file with three columns which contains the scores. The number of elements depends on the user. Once the program is executed, the output console asks the user to enter the value of $k$, after the value for $k$ is entered, the user is asked to enter the value for confidence. The range of confidence typically varied from 85 to 100 in order to find the results with higher accuracy and confidence levels. Depending on the tuples in the input file the tuples are first loaded and then sorted to calculate the top-$k$ results as explained in the above section.

### 3.2 Calculating Confidence

The project has two important calculations. The first one is to calculate the top-$k$ results and the second one is to assign confidence factor to the results. The above sections have clearly explained the process of generating the top-$k$ results. This section explains the concept of calculating the confidence which is the second major achievement of the project. After the input file is taken by the program the means of the tuples are calculated the Mean Threshold Values. A cumulative frequency of the overall score mean of the tuples is plotted. As this is the column
which is unsorted and these are the values compared with the Mean Threshold Values, which are sorted. The usage of calculating the confidence is to reduce the visits of the number of tuples. The cumulative frequency distribution represents the mean values according to the frequency of the overall score means. From the distribution graph the number of means higher than the highest Mean Threshold Value can be extracted. The problem is that where in the data list are these located. With the reference cumulative frequency percentage if the Mean Threshold value is compared with these tuples with the highest score we can easily find these data elements instead comparing with the next highest Mean Threshold Value. The cumulative frequency itself becomes the confidence percentage of the top-$k$ results generated decreasing the visits to huge number of tuples. The results generated may not be as accurate as generated by the regular TA as this procedure eliminates visiting some tuples and only few data elements are compared to generate the results. The results of decrease in the number of tuples can be seen in the following section 4. The graph shows a comparison for the same number of top-$k$ results to be found with the number of tuples that are visited by both the algorithms TA and TA-CLT.

As the data elements in the dataset considered in the experiments follow a Gaussian distribution a much accurate results can be calculated by finding the area under the curve for each Mean Threshold Values. The series is represented by the Eq. (3.1) below:

\[
 f_{\mu,\sigma}(x) = \int_{-\infty}^{\infty} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \frac{e^{-\frac{(x-\mu)^2}{2\sigma^2}}}{\sigma \sqrt{2\pi}} \quad (3.1)
\]
The area under the curve drawn from the above equation for each tuple with the standard mean and standard deviation gives the actual confidence percentage of each data element. In the above equation (3.1), ‘μ’ is the mean of the distribution and ‘σ’ is the standard deviation of the distribution. For a data element ‘x’ from the distribution, the area under the distribution curve gives the confidence of the element. The equations Eq. (3.2) and Eq. (3.3) are used to calculate the mean and standard deviation of a series with m data elements respectively.

\[ \mu = \frac{(d_1+d_1+d_2+\ldots\ldots+dm)}{m} \]  

(3.2)

\[ \sigma = \sqrt{\frac{\sum_{i=1}^{m}(d_i-\mu)^2}{m-1}} \]  

(3.3)
4 TESTING AND EVALUATION

The major purpose of the project is to reduce the time taken by the existing procedures to generate the top-\(k\) results. The evaluation can be done by considering the time taken to generate the top-\(k\) results by the method executed and the existing method. Various factors like the number of elements in the database and the range of Threshold Value should also be considered. Another purpose of this project is to generate fewer results with greater confidence. The project is tested with the outputs generated by the existing methods and the output generated by the algorithm implemented in this project.

4.1 Testing with various values of \(k\)

The project is tested with different values of \(k\) and a graph is plotted with various values of \(k\) ranging from 3 to 50. The time taken by the algorithm designed for the project and the existing algorithm are compared and plotted. The values are very close and the time taken is almost the same. The graphs are plotted for different range of data items from 100 to 100,000 rows. The time taken to find the top-\(k\) results by both methods is plotted and the resulting graphs show the efficiency and performance evaluation. During this testing the range of the tuples are constant and the value of \(k\) is changed from 3 to 50. The following figures from Figure 6 to Figure 11 are the time vs \(k\) comparisons for a fixed number of data elements.
Figure 6: Time in milliseconds vs. different values of $k$ for a database with 100 rows.

Figure 7: Time in milliseconds vs. different values of $k$ for a database with 1000 rows.
Figure 8: Time in milliseconds vs. different values of $k$ for a database with 5,000 rows.

Figure 9: Time in milliseconds vs. different values of $k$ for a database with 10,000 rows.
Figure 10: Time in seconds vs. different values of $k$ for a database with 50,000 rows.

Figure 11: Time in seconds vs. different values of $k$ for a database with 100,000 rows.
4.2 Testing with the range of data items in the lists

The results of the project are tested over a wide range of the data items varying from 100 to 100,000 items. For a same value of k the data range varying from 100 to 100,000 is tested with both the methods. A time - data range graph is drawn and the values are compares to evaluate the performance and efficiency of the existing method to that of the method implemented. This is a different view of the same data, the following figures from Figure 12 to Figure 17 shows the comparison of time taken for various data range for the same value of k. The x-axis is plotted with time in milliseconds and y axis with a data range of the data elements.

For $k = 3$

Figure 12: Time in milliseconds vs. number of data rows in a database for $k=3$. 
Figure 13: Time in milliseconds vs. number of data rows in a database for $k=5$

Figure 14: Time in milliseconds vs. number of data rows in a database for $k=10$
For $k = 20$

![Graph showing time in milliseconds vs. number of data rows in a database for $k=20$](image)

Figure 15: Time in milliseconds vs. number of data rows in a database for $k=20$

For $k = 30$

![Graph showing time in milliseconds vs. number of data rows in a database for $k=30$](image)

Figure 16: Time in milliseconds vs. number of data rows in a database for $k=30$
4.3 Evaluation

The Evaluation of the project is done by comparing the accuracy of the TA-CLT with the Threshold Algorithm and the number tuples seen by both the algorithms for generating the top-k query results. The algorithm which generates the top-k results with less tuple visits and less comparison between the Threshold value and overall scores is considered more effective and less time consuming algorithm. The following graphs are plotted with the number of tuples visited by both the algorithms, TA and TA-CLT, for a fixed number of data elements. The following graphs from Figure 18 to Figure 23 represent the comparison between the two algorithms based on the
number of tuples visited for various values of $k$ for datasets with rows 100, 1000, 5000, 10000, 50000, 100,000 respectively.

Figure 18: A graph showing the comparison between the tuples retrieved by both algorithms from a database with 100 rows and 3 attributes for various values of $k$
Figure 19: A graph showing the comparison between the tuples retrieved by both algorithms from a database with 1000 rows and 3 attributes for various values of $k$.

Figure 20: A graph showing the comparison between the tuples retrieved by both algorithms from a database with 5000 rows and 3 attributes for various values of $k$. 
Figure 21: A graph showing the comparison between the tuples retrieved by both algorithms from a database with 10,000 rows and 3 attributes for various values of $k$

Figure 22: A graph showing the comparison between the tuples retrieved by both algorithms from a database with 50,000 rows and 3 attributes for various values of $k$
Figure 23: A graph showing the comparison between the tuples retrieved by both algorithms from a database with 100,000 rows and 3 attributes for various values of $k$

From the experiments and testing done on TA-CLT, it is evident that the algorithm visits 10% fewer data elements than the Threshold Algorithm to generate the same top-$k$ results. As there is a decrease in the number of tuples to be visited, the time taken to calculate the top-$k$ results also decreases. Hence the TA-CLT is a faster processing algorithm than TA. As mentioned in the section 3 about the confidence with the cumulative distribution Figure 24 shows the cumulative distribution of the data elements. The following Figure 25 shows the comparison of the number of tuples against the various confidence percentages of the TA-CLT and the original TA. Figure 25 represents the experiment conducted on a data set with 5000 data elements and for $k = 50$. 
Figure 24: A cumulative frequency table for the data elements with the Mean Threshold Values.

Figure 24 is a cumulative frequency diagram of a data set with 5000 data elements. It shows the range for number of tuples with different values of means of the overall scores to find the top-\( k \) results.

Figure 25: No. of tuples visited by various confidence percentages of TA-CLT
TA-CLT also filters the results on the basis of confidence. This is an extra feature which TA does not have. The results can also be requested on the basis of the confidence level. It takes less number of tuples to calculate top-\(k\) results with less confidence so is the time taken. The test results are plotted accordingly by implementing CLT on the means of the score of the data elements on the three rows. The results generated are based on the confidence levels i.e. the area under the curve for each Threshold Mean Value. For 85% confidence, the numbers of tuples that have been looked up to generate the top-50 results are noted down. This confidence percentage is based on the area of the Mean Threshold Value is .85. The values generated for 80%, 90% and 95% respectively are noted and compared in Figure 25 with the original Threshold Algorithm. This particular experiment show that the TA-CLT algorithm is efficient than TA.
5. FUTURE WORK

The present system is developed based on the Threshold Algorithm. The algorithm generates a normal distribution graph which further verified with Kullback–Leibler divergence method could yield efficient results. However, there are many limitations for implementing Central Limit Theorem. As the values are random and not unique all the time in real time system, the graphs are plotted each time the results are generated. The plotting is done manually. An automation of drawing the graph could help in better performance. As the algorithm is based on Central Limit Theorem the results are rounded to the nearest percentage of confidence.

By implementing z-score calculator and standard deviation on the normal distribution graphs, generated by the Central Limit Theorem, the prediction range of the query results can also be calculated. The prediction of the results can be done only if the standard distribution and mean of the distribution is calculated. However a major drawback is that it cannot be implemented on a database which is a result of joining two databases. The scores or the ranking function might differ from one database system to the other. Hence, before the implementation of the algorithm, the scores of the attributes should be converted or must be ranked with a unique ranking function.
6. CONCLUSION

This project calculates the probabilities for the top-\(k\) results generated by the Threshold Algorithm using the Central Limit Theorem. This is a direct implementation of Threshold Algorithm over sorted data. This is a hybrid Algorithm combining two different techniques; one is used for generating results and the other for assigning probabilities. The efficiency and performance of the present system are better than the existing anytime algorithm version of Threshold Algorithm. The test results are compared and the comparisons are plotted as graphs which show that the project is efficient than the existing system. However the project is tested with an Excel file. In reality the project has the capability to work with database rows and tuples. An actual implementation of the project on a database will give a better estimation of the performance.

The main aim of the project is to design an algorithm for generating probabilistic top-\(k\) query results using Central Limit Theorem. The algorithm is designed and well tested with different range of data items and different values of \(k\). The project is implemented in c#. The test results and comparison graphs are plotted. The future work for the project and scope for development is well described. In spite of some limitations, the algorithm works efficiently for a very large range of data values. The project has served its purpose by yielding a better top-k query processing technique.
BIBLIOGRAPHY


