ABSTRACT

It is a recognized fact that humans usually tend to compare one thing with the other unless they get satisfied. During this process of expecting for the alternatives, they are noticing very much difficult to cipher out what they want. They typically look for the best one with more features, functionalities, good quality and at the same time something that is cheaper or affordable. Though the existing methods achieve high precision, they nonetheless suffer from low recall and performance issues.

The proposed system overcomes these drawbacks and improves the efficiency of mining entities. Therefore, in order to assure high precision and high recall, we prepare a novel approach that combines the Bootstrapping and relation keyword query questioning process for text search techniques in SQL. The primary goal of this project is to identify whether the question is comparable or not and then we pull out the required entities from the query by making use of an extensive online question record [1]. Initially, the user presents a query as an input; later the system will identify whether the passed question is comparable or not. Once the system verifies that the query given by the user is equivalent, the required entities are extracted, and the output is presented to the user with the possible options. This approach provides better results compared to the existing approach.
# TABLE OF CONTENTS

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

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

List of Figures ............................................................................................................ vi

List of Tables ............................................................................................................. vii

1. Background and Rationale ...................................................................................... 1

   1.1 Introduction ...................................................................................................... 1

   1.2 Information Extraction System .......................................................................... 5

       1.2.1 Rule-Based Extraction ............................................................................... 6

       1.2.2 Pattern-Based Extraction .......................................................................... 6

       1.2.3 Supervised Learning .................................................................................. 7

   1.3 Overview of Data Mining .................................................................................... 7

   1.4 Existing Comparative Mining Systems .............................................................. 8

       1.4.1 Comparative Mining Based on Entity Disambiguation with Markov Logic Network ................................................................................................................. 8

       1.4.2 Clique Grow: Mining Entities from the Network ............................................ 9

       1.4.3 Comparable Entity Mining using Bootstrapping and Apriori TID Algorithm ................................................................................................................................. 9

   1.5 Literature Review .............................................................................................. 11

2. Narrative ................................................................................................................ 14

   2.1 Problem Statement .......................................................................................... 14
2.2 Motivation........................................................................................................14
2.3 Project Objective..........................................................................................15
2.4 Project Scope...............................................................................................16
3. Proposed System Design................................................................................17
  3.1 System Design and Architecture...............................................................17
    3.1.1 Bootstrapping......................................................................................17
    3.1.2 K-Nearest Neighbor...........................................................................22
    3.1.3 K-Nearest Neighbor Algorithm.........................................................24
  3.2 Parser-Chunking Tagger.............................................................................25
3.4 Environment..................................................................................................26
  3.4.1 Java Server Pages..................................................................................26
  3.4.2 Java Servlets.......................................................................................26
  3.4.3 JDBC.....................................................................................................27
  3.4.4 NetBeans IDE 7.0.1.............................................................................27
  3.4.5 JDK 1.7.................................................................................................28
3.5 System Requirements..................................................................................29
  3.5.1 Software Requirements.........................................................................29
  3.5.2 Hardware Requirements.........................................................................29
4. System Implementation....................................................................................30
  4.1 User Interface.............................................................................................30
  4.2 Data Retrieval............................................................................................33
  4.3 Parser-Chunking Mechanism......................................................................36
  4.4 Comparator Extraction..............................................................................39
4.5 Patterns Evaluation..............................................................................39

5. Testing and Evaluation........................................................................40

  5.1 System Testing..................................................................................40

  5.2 Unit Testing.....................................................................................40

  5.3 Integration Testing ..........................................................................41

  5.4 Functional Testing ...........................................................................41

  5.5 Experimental Evaluation...............................................................41

  5.6 Test Cases.......................................................................................49

6. Conclusion..........................................................................................58

Bibliography and References.....................................................................59
## LIST OF FIGURES

| Figure 1.1 | Difference Between Traditional Versus Entity Search | 2 |
| Figure 1.2 | Example of Traditional Search Results Retrieval | 4 |
| Figure 1.3 | Information Extraction Architecture | 5 |
| Figure 3.1 | System Overview | 20 |
| Figure 3.2 | Bootstrapping and K-Nearest Neighbor Architecture | 21 |
| Figure 4.1 | User Interface | 30 |
| Figure 4.2 | Login Access | 31 |
| Figure 4.3 | Uploading Product Related Datasets to the Database | 32 |
| Figure 4.4 | Image Upload | 33 |
| Figure 4.5 | User Input Query | 34 |
| Figure 4.6 | Results of Comparator Mining | 35 |
| Figure 4.7 | Results of Comparator Mining | 36 |
| Figure 4.8 | Parts of Speech Identification | 37 |
| Figure 5.1 | Login | 50 |
| Figure 5.2 | Successful login | 51 |
| Figure 5.3 | Invalid Input Query | 52 |
| Figure 5.4 | Error Results of Invalid Input | 53 |
| Figure 5.5 | Input Query with no Matching | 54 |
| Figure 5.6 | No Matching Result Returned | 55 |
| Figure 5.7 | End User Input Query | 56 |
| Figure 5.8 | Relevant Results Retrieved | 57 |
LIST OF TABLES

Table 3.1 Data for the Training Vectors .................................................. 23
Table 3.2 Data for the Feature Vectors .................................................. 23
Table 4.1 Parser-Chunking Tagset ......................................................... 38
Table 5.1 Results Table ........................................................................ 45
Table 5.2 Performance Measurements .................................................. 46
Table 5.3 Results Returned by the Existing System, Proposed System
And Google Search ............................................................................. 47
1. BACKGROUND AND RATIONALE

1.1 Introduction

Comparing alternatives are a crucial step in human decision-makings. For example, someone is interested in cars, mobile telephones, desktops, perfumes, cameras, laptops and so on; they would wish to recognize what the choices are and how can they compare each other, of those options before arriving at a decision.

This medium is often common in our day to day life and demands a high achievement. Many websites such as Engadget, 9to5mac, Life Hacker, CNet.com and The Verge, strive very hard to provide their editorial choices for comparison. The comparison involves looking for relevant web pages that contain the info about the products what they are seeking for, user reviews, distinguishing advantages, and disadvantages [1]. Mostly, they make an overview of the end product that the user is looking for. Therefore, in this project, we focus on creating a set of comparable entities by making use of the end user’s input [1]. For example, we are looking to buy Samsung Galaxy S5, and we are awaiting for the comparable alternatives such as the iPhone 6, Lumia 600 and so on so that we can get the best for what we are paying for. Things get quite complicated here because of the rationality that one is of the lower end model with fewer features and functionalities, and the other is a higher goal with more functionalities and characteristics. For instance, one might compare “iPhone” and “Nintendo Wii” as portable gaming devices that were right to an extent. Though iPhone provides gaming functionality, it is mainly used as mobile device for making phone calls, sending messages and hence on. Gaming feature is a part of the iPhone, but it is not entirely
dedicated as a gaming console. On the other hand, one can consider “Wii” as an entirely dedicated gadget for gaming. For this cause, thousands of questions related to the comparison between gadgets, automobiles and so on are posted on the internet for which many of them are yet unrequited. Here “Wii”, “iPhone” are called as comparators or entities.

The primary object of this paper would be first to identify whether the question presented by the user is comparable or not and then mine the necessary entities from queries and deliver the user with the alternatives. This medium would aid finding the options very easy [1]. A question that does not take in any intent to compare things would not be called as a comparative inquiry. Figure 1.1 [18] shows the difference between the traditional search and entity search.

![Figure 1.1. Difference Between Traditional Versus Entity Search](image-url)
Figure 1.1 depicts the traditional approach where we are unable to receive exact and relevant information when we tend to look for the things over the internet. By using this approach, we can extract only a few things related to the end user query from the online web sources. For instance, when a user gives a query to the search engine like ‘Google’ it returns the results from several web pages which usually are enormous and needs a lot of manual effort from the end user to go through each web page separately to figure out what is the best choice for them. Figure 1.1 depicts the entity search, we can achieve the same alternative choices which the end user is looking for without going through each and every web page manually and figure what’s best for them. By using this approach, we can extract many relevant things related to the end user query from the database. For instance, when a user inputs a query “Which is better, Nikon D600 or Sony s550?” The system will return the relevant alternative choices to that end user straight away without making the user go through web pages and figure out manually what’s best for him which is very tedious and time-consuming. In entity mining, we do keep a database of the relevant domain what the terminal user is counting onward to make things simple so that we can retrieve the information at a faster pace.

For instance, Figure 1.2 represents the results retrieved by a conventional search engine in response to the query “Which is better iPhone 6 or Galaxy s5?” Equally, we can regard the solutions yielded by the search engine are enormous, and the terminal user has to move manually through each document to determine the relevant option. So, the information extraction system processes the user question and provides the user with the possible alternative options.
Figure 1.2 Example of Traditional Search Results Retrieval.
1.2 Information Extraction System

The procedure of extracting organized pieces of data from any disorganized or semi-structured documents are called as Information Extraction [2]. The chief goal of Information Extraction is to cater the end users with a meaningful slice of data which they can use it based on their demands. The extraction includes people, businesses, and places so on. In order for us to extract this meaningful information, we need to make use of an Information Extraction concept. An important approach called as Entity Recognition is used to detect and categorize any data in the given piece of textual matter. Based on the extraction types, there are three methods that are commonly used for retrieving the relevant data. They are Rule-based Extraction, Pattern-based Extraction and Supervised Learning [2]. Figure 1.3 [19] shows the Information extraction architecture.
1.2.1 Rule-Based Extraction

This method is based on spontaneously learning the schema related mining rules in order to detect each possible type of connection or objects. In this approach, the patterns are predefined and are utilized only when enough training data is not available or when the domain the end user is trying to implement is very unmanageable to be taken care by automated methods. For instance, consider the model introduced by Rapier in [20] where patterns are represented in a more improved steady formula language. In summation to this a rule learner based on bottom-up comparative is employed to implement rules from a corpus of categorized training instances [20]. A new language called as Inductive Logic Programming (ILP) [21] was used in order to learn logical rules to identify and extract expressions from a text file. This approach is no longer used because of its poor performance issues.

1.2.2 Pattern-Based Extraction

This approach was developed on marked text patterns [2]. The fundamental idea was to label pieces of information such as keywords or expressions with verbal information. It consists of Part-Of-Speech Tagger or syntax-based information. Subsequently, in order to discover the existence of the connection they make use of the patterns to verify against the verbally marked text [2].
1.2.3 Supervised Learning

In Supervised Learning, we train the system as much as possible by tagging the data so that the system would be able to carry out the job [2]. For illustration, take the face recognition system where we require labeling the faces manually so that the system would be able to detect automatically and observe the same face name next time when it shows up. This approach works based on the previous data. Without this past data, the system will not function anymore as it would not have any idea of what it is doing. The drawback of this approach is that it is very time-consuming and expensive as it takes a great deal of manual training by the user.

1.3 Overview of Data Mining

Data Mining is a method of learning knowledge from voluminous amounts of databases [33]. The critical aspect of data mining is to collect and store the data that is later used for several purposes. During this process, the stored information is translated into operational data to conform to the demands [33]. It serves a potent instrument for decision-making for various lenders and commercial enterprise. For example, people use their credit card for making a hefty sum of transactions where the financial lenders make use of automated mining systems to detect the presence of a fraud activity going on. Several algorithms are used to mine the information based on the needs. For example, American Express was using its customer’s transaction history to discover an unusual spending pattern called as a profile behavior to reduce the credit limits of their clients to shorten the hazard level between the customer and the loaner. Likewise, various departments and food market shops are using the Client initiated transactions, historical
data to settle what the customers buy the most and what they dislike so that they can amend their business needs as per the user demands.

1.4 Existing Comparative Mining Systems

Several existing comparative mining based systems are discussed in brief in this section

1.4.1 Comparative Mining Based on Entity Disambiguation with Markov Logic Network

This approach is based on the aggregation of the first-order logic (FOL) combined with NIL-filtering and entity disambiguation stages [3]. An NIL-filtering can be used to pair facts that are reported as nil. A background database is maintained, which contains the glossary of all possible entities. For example, when somebody poses a question the system maps the query to the database and calls back the required objects. The formulations are inferred based on four keywords that include constants, variables, functions, and predicates [3]. Constants refer to objects in the repositories; variables are denoted by the unknown variables such as P, Q for the appropriate entities and the predicates are used to build a link between these entities. The information contained in the database maybe authentic, corrupt or unidentified.

Disadvantage

The main downside of this method is that it is very complex and needs more time to process questions and is entirely dependent on the background knowledge otherwise it
would not be able to find relevant answers. In addition, the system doesn’t maintain any filtering mechanisms to do away with corrupt or unknown data.

1.4.2 Clique Grow: Mining Entities from the Network

For a given entity graph, this approach is applied to discover missing links by making use of query logs [4]. The primary difference between Clique Grow versus the existing arrangements, such as Markov Logic, Bootstrapping and Apriori is that the Clique Grow does not extract only the entities that are compared in the corpora, but it has also drawn out the ones that occur very rarely [4]. This approach includes all the entities making it a complete system. Here we expand the known comparable relations to unknown relations. It makes use of a graph and clustering concept.

*Advantage*

The main advantage is that this approach is capable of solving the ambiguities between the entities present in two or more comparator pairs.

1.4.3 Comparable Entity Mining using Bootstrapping and Apriori TID Algorithm

This approach makes use of bootstrapping as well as the apriori algorithm [31]. Unlike other systems, when the user posts a query the mechanism is divided into two stages in order to obtain the entities. Initially, when the question is posed the Bootstrap algorithm verifies whether the passed question is comparable or not. In one case, it declares the question is comparable then it is given to the Apriori TID where its functionality is to pull out the comparable entities and deliver the possible alternative options to the final stage user [31].
Advantage

This approach achieves high recall and precision, unlike other systems.

1.4.4 Mining Comparative Sentences and Relations

This method proposed by Jindal and Liu makes use of the concept of class consecutive Rules (CSR) and label consecutive Rules (LSR) [32]. This approach works by mapping an input sequence pattern that consists of keywords, parts of speech or variables to a labeled sequence consisting of labels [1].

Disadvantage

However, there are few drawbacks to this method. Firstly, though this method achieves high precision it still suffers from low recall. Secondly, the existing method makes use of predefined keywords, about 83 which are likely to serve as indicators of comparing sentences [32]. Nevertheless, the performance is a massive event as it relies heavily on those manually created keywords as they offer no proper instructions on how to select those keywords. If we run into any other comparative question outside these 83 sets of keywords, the system wouldn’t be applicable any longer. Moreover, when users are putting questions in several different ways it constructs the system very complex as it has to be cultivated to learn which consumes a lot of imaginations, and consequently it becomes not a cost effective method.
1.5 Literature Review

Several researchers have made a lot of contribution to the field of entity mining. Newcombe et al. [5] was the first person to define entity matching that was later validated by Felligi and Sunter [22]. Ha [6], [7] sorted out the format of Korean comparison sentences into several classes and organized contrast related keywords from a semantic outlook.


Leitao et al. [12] proposed a framework that does not utilize trained data. This method was applied to real and the artificial problems that include videos and compact disc. The primary focal point was on pairing XML entities grounded on a Bayesian Network. A graph dependent pattern is provided by default in order to declare the relationship among the entities of a domain. The terminal user is supposed to fix a threshold where all the entities above the preset threshold are counted as equals.

A. Thor and E. Rahm [13] proposed a framework called MOMA (Mapping-based Object Matching), which is independent of the domain provides a vast array of matchers. It consists of the attribute as well as context matchers. They made use of a relational entity type that utilizes the concept of workflow to ensure that mappings occur between...
the entity types. Additionally a storage is included to store the necessary mappings and can be practiced once again in other workflows. It makes use of the offline data. It was used to test the real problems that include publications, authors, and venues.

O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S.E. Whang, J. Widom and Swoosh [14] proposed a framework called SERF (Stanford Entity resolution Framework), which has a generic base. The primary focus was placed on improving the efficiency of the matching. This was employed to prove the actual problems that include hotels and products. They made use of Trigram and neighborhood matchers and several algorithms to scale down the count of requests by keeping track of previous values.

Tejada et al. [15], [16] Proposed Active Atlas system that makes use of the decision trees to train the system. They made use of a semi-automatic approach to minimize the count of training examples. It was applied to actual problems that include Restaurants, Companies and weather traffic data. A variant of TF-IDF matcher was used.

Chaudhuri et al. [17] proposed the concept of operator trees that includes a vast assemblage of various matching joins. An algorithmic divide and conquer approach were applied to build those operator trees by making use of offline data. Users hold the power to limit the count of matching links. This was applied to actual problems that include companies, persons, publications, restaurants, and birds.

Kiran Kumar Velineni [23] proposed semantic QAS, which provides responses for “Who is” queries using semantic cloud tags. The central idea was to process to recognize the sentences that contain semantically related query keywords and to focus the importance of semantic processing.
Deepti and Prashant [24] proposed a weakly supervised approach where it makes use of an input sequence mapped to a labeled sequence by replacing the tokens in labeled with the input. Their approach is almost similar to J&L’s method where rather than using rules they were using sequential patterns. The drawback with this approach is that it relies heavily on keyword indicators causing performance issues. Moreover, it doesn’t extract the rare mining patterns.

Pavlos and Yannis [25] proposed an integration concept focused primarily on improving the results from non-semantic search systems that include professional and regular web search engines. A link analysis based approach is applied to depict the ranking in search outcomes and to generate K-semantic graphs [25]. By using this, the users can locate the information spontaneously residing in various positions related to the entities.

Kai-Sheng, Chun-Cheng and Yuen-Hsien [26] proposed entity mining concept based on part-of-speech tagging. This methodology was used to cipher out the criminal acts, litigation information and investigation related clues by the law enforcement team. In order to achieve this, a network is built for entity related visualization and exploration. The network includes mappings between various criminals and their related activities.

Zaiqing, Ji-Rong and Wei-Ying [27] proposed a new methodology called statistical web entity extraction where the central idea is to dig up and aggregate all the web related data regarding an entity together as information unit. They made use of vision-based web information that includes layout and stand structures to see out the skeleton and content of the web pages.
2. NARRATIVE

2.1 Problem Statement

Decision-making is a challenging task for anyone. The traditional search engines haven’t been amended thus far where the end user would be able to extract the possible entities that they might be fronting for. When it comes to the concept of entity extraction, we need to make certain that we can reach very high recall and precision that is a central component. Precision refers to the ratio of the number of correct answers obtained to the total number of results retrieved [35]. Recall refers to the ratio of the number of correct answers obtained to the total number of relevant results [35]. Though the existing methods achieve high precision, nonetheless suffer from low recall. These current methods rely on manually predefined keywords about 83 in order to distinguish whether a given sentence or a question is comparable or not. Because of this performance is a huge drawback as it is largely based on these keywords. Secondly, when the user poses any question apart from these keywords, they system wouldn’t be applicable anymore to detect the comparable sentence. Aside from these most of the time they represent errors. Moreover, these existing methods are entirely hooked on the Structured Query Language making the overall execution slow.

2.2 Motivation

People usually have a tendency to compare to other alternatives when they are looking to buy something. They would like to find the one that has more features, functionalities and at the same time, something that is affordable to their everyday needs. Thus, in order to obtain the best product people several post thousands of queries relating
to them on various online sites, blogs and so on. Most of the questions remain unanswered for these queries. For example, consider the traditional search engines when we attempt to ask a question it shows the relevant internet sites associated with those queries, but it does not directly present the result to the end users providing possible alternatives. It is very crucial for the people to choose the correct one among the possible alternatives they have got before they purchase the wrong one. Thus providing the potential alternative options to the user for their given query would be an accomplished work.

2.3 Project Objective

Considering the existing systems and their drawbacks we are attempting to go through a novel scheme which would be capable to overcome the issues the earlier systems had and try to improve the overall efficiency, execution speed and the F1-Measure [1] of the system. F1-Measure refers to a system that produces both high precision and high recall that indicates the proposed system is very well efficient and faster compared to the previous ones. In this task, we are attempting to put through a hybrid approach that is the combination of Bootstrapping as well as Relational keyword query questioning process for the text search techniques in SQL. The relational keyword query concept makes use of the K-Nearest Neighbor approach [34] that improves the overall execution speed and would be able to achieve high precision and high recall which we are fronting for. We would improve the overall mining outline of the entities and excavate exceptional mining patterns.
2.4 Project Scope

The central goal of the task is to provide the users with the possible alternatives for their queries so that the users can make the correct selection of the products what they are looking for in any particular domain satisfying their needs. People usually tend to post several online questions on several search engines and weblogs in order to screen out the best outcome what they are waiting for. This procedure is very complex, and the user will not be coming out with the possible alternative options. We make use of the online data records usually the questions sent by these users online and provide the alternatives what they are looking for straight away.
3. PROPOSED SYSTEM DESIGN

3.1. System Design and Architecture

Initially, before the application is made accessible to the end users, the administrator will upload datasets about the relevant domain. This includes all the relevant products and information about them. In this project, we create a shopping website where the terminal user is given the access to browse the list of items available on the website before they arrive at a conclusion. This provides the end users a better characterization of what they are looking for in order to make their alternative choices and arrive at a concluding determination.

Figure 3.2 [1] shows the entity mining process architecture. In this project, we are going to implement a bootstrapping approach [1] and a K-Anonymity based approach called Nearest Neighbor. Figure 3.1 depicts the system overview.

3.1.1 Bootstrapping

Bootstrapping is a procedure for extracting the alternative results for a given user query. It consists of query identification, noise filtration, and comparator extraction. For instance, when an end user presents a query to the application, in order for the system to extract the alternative results based on the user query, it needs to go through the bootstrapping procedure. Firstly, the end user goes to the online website to catch a wider understanding of what they are looking for before they present a question to the system. Later, the user submits a query to the system. In the query identification phase, the query posed by the end user is passed through the Parser-chunking tagger where two
mechanisms take place. Initially, it identifies whether the given end user query is a comparative question or not. In order to identify whether the given query is a comparative query or not we make use of adjectives such as ‘better’, ‘prefer’ and so on. If it is successful in identifying the query to be comparative then the control is passed to the next phase ‘Noise Filtration’ where all the noise is removed and the comparator pairs are extracted. Here, the comparator pairs are the user given products. The noise includes all the parts of speech related keywords in the query such as he, she, was, better, than, this, that and so on. Once the entities are extracted from the input query the bootstrapping approach now needs to extract the relevant alternative queries for the end user. In order to achieve, initially it checks the existing cache to identify any relevant matching comparators that may exist for that query. If it is unable to trace any results for that query from the cache, then the control is passed on to the database where the comparison is done to find the relevant alternatives and the results are displayed to the end user.

During this process, it makes use of a cache that is available in the database so that the results are retrieved at a faster rate. The cache contains the history of all the previous search queries made by the users. By doing this, the comparison time is reduced. Since the data are of cache nature existing in the database, all the results are retrieved by the index.

Figure 3.2 depicts Reliable seed pairs consist of comparator pairs that are of highest rating, and the question archive has cache memory in the database for the bootstrapping mechanism.
Based on the information available in reliable seed pairs and question archive we generate pair candidates, which are the alternative choices the end user is looking for. Whenever an end user clicks the alternative choices, they are considered of highest rating, and are added back to the reliable seed pairs. The IEP is a database consisting of all the product related information. This is a learning based approach where the system tries to learn every time the results are retrieved and when the user clicks those alternative choices.
Figure 3.1 System Overview
Figure 3.2 Bootstrapping [1] and K-Nearest Neighbor [34] Architecture
3.1.2 K-Nearest Neighbor

In this approach, we maximize the efficiency performance metrics of the system by relying on the key insight that finding the nearest neighbor approach produces the most accurate results. Initially, once the input query is filtered by using the parser-chunking mechanism all the relevant noises are removed and the extracted comparators are passed to the nearest neighbor approach.

In K-Nearest Neighbor approach, we initially define P training vectors, and the algorithm identifies the K nearest neighbors for the feature vector class C. Labels are associated with the training and feature vectors. Here the P training vectors refer to the existing products related datasets available in the database, C is the feature vector for which the nearest neighbors have to be determined and K is a positive integer that is user-defined. In order to restrict the ties among the data, we usually consider odd values for K where K=1, 3, 5…… n. To determine the nearest neighbors for the Class C based on the K value, we need to make use of a mathematical distance function which can measure the distances between the training vectors and the feature vector. In this application, we made use of the approach ‘Euclidean Distance’ [34]. It is defined by the mathematical formula

$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

[34] Where Xi, Yi are referred as the distance parameters. This formula is used to measure the distance between points. In order to find the nearest neighbors for a given query, the key idea is to compute the distance between the feature vector C and the training vectors. For instance, consider Table 3.3 which shows the training vector datasets and by using that data the Euclidean distance is calculated, and the class for feature vectors are retrieved.
Table 3.1 Data for the Training Vectors

<table>
<thead>
<tr>
<th>Products</th>
<th>Processor</th>
<th>Screen size</th>
<th>Ram</th>
</tr>
</thead>
<tbody>
<tr>
<td>iPhone</td>
<td>1.5</td>
<td>4.7</td>
<td>1.8</td>
</tr>
<tr>
<td>Galaxy S6</td>
<td>2</td>
<td>5.5</td>
<td>2</td>
</tr>
<tr>
<td>Lumia</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3.2 Data for the Feature Vectors

<table>
<thead>
<tr>
<th>Products</th>
<th>Processor</th>
<th>Screen size</th>
<th>Ram</th>
</tr>
</thead>
<tbody>
<tr>
<td>M8</td>
<td>1.8</td>
<td>5</td>
<td>2.5</td>
</tr>
<tr>
<td>Classic</td>
<td>1</td>
<td>3</td>
<td>0.7</td>
</tr>
</tbody>
</table>

For instance, when the end user gives the query “which is better m8 or classic”. The Euclidean distance is determined between the feature vectors m8, classic and each of the training vectors iPhone, Galaxy S6 and Lumia. Table 3.4 depicts the future vector product related attribute data for M8 as: processor is 1.8, Screen size is 5, and Ram equals 2.5. Now we calculate the Euclidean distance as follows \[ \text{Sqrt} \left( (1.8-1.5)^2 + (5-4.7)^2 + (2.5-1.8)^2 \right) \] which equals 0.58. Similarly, the Euclidean distances for M8 and Galaxy S6 is calculated as \[ \text{Sqrt} \left( (1.8-2)^2 + (5-5.5)^2 + (2.5-2)^2 \right) \] which equals 0.54 and for M8 and Lumia it equals 3.89. In the similar way, the distances for feature vector
classic is determined. Likewise, we calculate the distances for all existing products in the database and their computed distances are used to determine the nearest neighbors.

Based on the K value if it equals three which means three nearest neighbors have to be retrieved for the class C. The three lowest values of the computed Euclidean distances are extracted closer to the feature vector attribute value, and their relevant product related information is displayed to the end user. It is recommended to use higher K value so that more relevant results are going to be generated with low noise.

We then also create a view based approach where we try to display all the products that have the highest rating as a first come output and rest of the products secondary based on the number of hits made by the end users. These hits are triggered when the end user clicks on the alternative options retrieved. The hits are usually incremented as long as the end users retrieve the alternative products and select them. Therefore, this is considered to be of trustworthy and highest rating.

3.1.3 K-Nearest Neighbor Algorithm

1. Define the constant K which equals the number of nearest neighbors.

2. Compute the distance between the feature vector and all the training vectors by using Euclidean Function.

3. Sort the distances for all the training vectors and determine the nearest neighbors based on the K-th minimum distance.
4. If $K=1$ then the feature vector is assigned to the class of its nearest neighbor.

5. If $k>1$ then select the nearest neighbors for the feature vector class based on its attribute value closer to the lowest Euclidean distance values obtained by computing the distance between all the training samples and for that feature vector.

6. If necessary, collect the relevant category $M$ related to the nearest neighbors.

3.2 Parser-Chunking Tagger

In this project, we make use of a parser-chunking tagger an API that will distinguish the components of language keywords in the given input query by the terminal user. These keywords serve an indicator for identifying whether the given query is comparable or not. By making use of this, the system can determine whether the given query is a comparative question or not. If the end user is given only the comparators without giving any parts of speech keywords, then it would not be considered a comparative question and the system will not go ahead retrieving the answers for the final stage user. For instance, “which is better Apple or Samsung”. When the end user puts this question into the system, the parser-chunking tagger first scans the question completely distinguish any portions of the speech related keywords like Noun, Pronoun, Adjective, Verb and hence along. It is starting to classify “Apple, Samsung” as Nouns that are the comparators, better as the adjective and so on


3.4 Environment

3.4.1 Java Server Pages (JSP)

Java Server Pages is a technology that is widely embraced by the developers to control or alter the network pages. This is achieved by making use of servlets which are based on server-side technology. Servlets are programs that reside in the web pages and modify the page before the access is authorized to the user who initially requested it [28]. They consist of two types of data. The foremost single is called Static which is of a usual text based and the later one called as JSP element. This is referred by JAVA as the Servlet API. An activity is initiated before the page is sent to the end user. In that respect are various benefits of using these server pages. These provide robustness for the web applications where end users and developers heavily rely on it.

i. Developers can practice this without having any good amount of knowledge about Java.

ii. This can be prolonged for various other applications including APIs for various web applications.

iii. It has inbuilt tag library to back up various functions that can be practiced for most of the time while designing web applications.

3.4.2 JAVA Servlets

These are small programs residing within the host. They usually tend to respond to the requests of the terminal user via HTTP. Initially, when needed, by the user these are constructed and initialized for usage. Later all the requests from terminal users to the servlets are handled by call routines. In the end when these are no more involved, they
are destroyed by the user to amend the performance and fix the quantity of memory space the program resides.

### 3.4.3 JDBC

This presents as an interface or connectivity between Java dependent applications and various other databases [29]. This includes spreadsheets, flat files, and other transaction systems including enterprise resource planning. This is commonly referred as JDBC API which contains in an advantage of portability where after writing the code for once can be executed anywhere. It delivers unique advantages which include:

i. Business can make use of the existing data which is present in several database management systems in a remote access mode.

ii. The API is simple to read, understand and deploy.

iii. On the client side, no installation is needed everything can be accessed by the workspace URL.

iv. There is no restriction on the data access created by the developer.

### 3.4.4 NetBeans IDE 7.0.1

This project makes use of NetBeans IDE 7.0.1. This is an open source IDE tool available for the public to use it. This application is used in designing various desktops, mobile, web, HTML, PHP, and C/C++ applications [30]. This can be run on Windows, Mac, Linux and Solaris operating systems. The workspace environment is very much more flexible than a regular text editor, providing several functionalities such as code indentation, templates, and tips.
It provides the alternate views of data which includes folder and hierarchy, allowing the developer drill down to the actual source of data spontaneously [30]. It also provides GUI builder, which automatically takes care of all the necessary formatting and alignment [30]. It holds a very bug free environment where specialized debugger takes care of all the bug related issues, offering an in-depth information about the error causing the end user set up the subject very quickly.

3.4.5 JDK 1.7

This is made available to the public as an open source tool that is utilized to run Java applications and applets. The JDK consists of the inbuilt features such as Java runtime environment, compiler, and the APIs. This can be configured to install and function on the platforms such as Windows, Mac, Solaris, and Linux.
3.5 System Requirements

The minimum software and hardware requirements for the application are:

3.5.1 Software Requirements

• Language : Java
• Version : JDK 1.7
• IDE : Net-beans IDE 7.0.1
• Database : Oracle 11g r2

3.5.2 Hardware Requirements

• Processor : PENTIUM IV
• Clock Speed : 2.5 GHZ
• Ram Capacity : 1 GB
• Hard Disk Drive : 250 GB
4. SYSTEM IMPLEMENTATION

4.1 User Interface

The web interface consists of the login access to the end users as well as the administrator of the application. It also comprises the search bar where the end user inputs the query for retrieving their alternative choices. Moreover, it exposes an entire list of all the products uploaded by the administrator to the database. From here, the end user accessing the application owns the access to view entire products available in the database. This makes it very easy for the end user who is unsure of what to buy. Figure 4.1 Shows the web interface.

![User Interface](image-url)

Figure 4.1 User Interface
Figure 4.2 depicts the login access for both the end users as well as the administrator. The end users can select the new user option and register as a first time user.

![Login Access](image)

**Figure 4.2 Login Access**
Once the admin has logged into the application, the admin can update the datasets in the database as well as mention all the related attributes of the product such as brand name, model, price, screen size, ram, processor, internal memory and external memory. Afterward, all the details are given so that the administrator can append it to the database. However, the administrator of the application has the access to upload the product related, datasets to the database. The end user is restricted from making any changes to the database. The fig 4.3 shows the uploading data to the database.

![Figure 4.3 Uploading Product Related Datasets to the Database.](image-url)
Once all the product related information has been updated to the database, the administrator then uploads all the relevant images of each individual product. The images are uploaded based on the unique id generated for every product. The figure 4.4 depicts the image upload.

![Image of Mobile Shopping Zone](image)

**Fig 4.4 Image Upload**

### 4.2 Data Retrieval

Once the relevant product related information has been updated to the database the end user tries to look for the alternative choices for his query by entering it in the search bar of the application. Once the query has been submitted, the results are retrieved by using the parser-chunking technique and Comparator algorithm. All the results include
the relevant products related to the input query. The figure 4.5 shows the end user input query.

![User Input Query](image)

**Fig 4.5 User Input Query**

The results are shown in figure 4.6 and 4.7 which contains all the attributes of the products along with the product information.
<table>
<thead>
<tr>
<th>Brand Name:</th>
<th>Samsung</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model:</td>
<td>galaxy note</td>
</tr>
<tr>
<td>Price:</td>
<td>300.0</td>
</tr>
<tr>
<td>Screen size:</td>
<td>5.5</td>
</tr>
<tr>
<td>Processor:</td>
<td>2.0</td>
</tr>
<tr>
<td>Ram:</td>
<td>1.0</td>
</tr>
<tr>
<td>Internal Memory:</td>
<td>16.0</td>
</tr>
<tr>
<td>External Memory:</td>
<td>64.0</td>
</tr>
<tr>
<td>Hits:</td>
<td>18</td>
</tr>
</tbody>
</table>

**Fig 4.6 Results of Comparator Mining**
4.3 Parser-Chunking Mechanism

Once the query is given to the application, initially it is redirected to the parser-chunking mechanism where the user query is scanned thoroughly to identify whether the given question is a comparative question or not. In order to achieve this, the parser tagger identifies all the relevant parts of speech tags in the input query and based on those identifications it detects the query is a comparative one and pass through the next phase of comparative mining. By making use of the adjectives such as ‘better’ it is making a
decision whether the query is a comparative or not and should be passed to the next phase for further processing or not.

List of Noun parse : [porsche, Lumia]
List of Adjective parse : [better]
List of Verb parse : []

Figure 4.8 Parts of Speech Identification

Figure 4.8 demonstrates how the system identifies parts of speech in a given query “which is better Porsche or Lumia” based on the parser-chunking mechanism. Here ‘Porsche’ and ‘Lumia’ are classified as Nouns and ‘better’ as the adjective which identifies the query to be a comparative one. This approach makes use of different kinds of parts of speech available in the Parser-Chunking API. For instance, Nouns exist as singular noun or plural noun, verbs, and other adjectives exist in the similar way. Table 4.1 below lists the parts of speech tags.
<table>
<thead>
<tr>
<th>POS Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN</td>
<td>Noun, singular</td>
</tr>
<tr>
<td>NNS</td>
<td>Noun, plural</td>
</tr>
<tr>
<td>NNP</td>
<td>Proper noun, singular</td>
</tr>
<tr>
<td>NNPS</td>
<td>Proper noun, plural</td>
</tr>
<tr>
<td>JJ</td>
<td>Adjective</td>
</tr>
<tr>
<td>JJR</td>
<td>Adjective, comparative</td>
</tr>
<tr>
<td>JJS</td>
<td>Adjective, superlative</td>
</tr>
<tr>
<td>VB</td>
<td>Verb, base form</td>
</tr>
<tr>
<td>VBP</td>
<td>Verb, non-third person singular present</td>
</tr>
<tr>
<td>VBG</td>
<td>Verb, present participle</td>
</tr>
<tr>
<td>VBD</td>
<td>Verb, past tense</td>
</tr>
<tr>
<td>VBN</td>
<td>Verb, past participle</td>
</tr>
</tbody>
</table>
4.4 Comparator Extraction

Once the query is filtered, the comparators are identified, and the control is passed to the comparator algorithm where the application tries to look for the comparators in the cache first. If they are found in the cache, the results are retrieved at a quicker rate, and they are displayed to the end user. Otherwise, when they do not exist in the cache they are retrieved from the database, and the comparison is triggered to find the alternative options for the user query. The results are added to the cache whenever the end user clicks on the product to see them after the results are retrieved. This usually increases the hits of the individual product so that next time, when the end user tries to obtain a similar product the products which contain more hits, are retrieved first and the others are shown below it.

4.5 Patterns Evaluation

In order to provide more flexibility for the end users, we can create n number of patterns making the queries meaningful and can be understood by every other user. For instance “Which is better $C or $C?” is a rule that provides meaningful data when the end user is attempting to retrieve his alternative comparators. Here $C, $C refer to the comparators that are the products the end user is seeking help for.
5. TESTING AND EVALUATION

5.1 System Testing

Testing involves detecting the various bugs within the system. It is usually performed to build sure the end product is working accurately without any variety of flaws or glitches. By doing this testing, the developers can assure the end users that the product is working precisely and can be saved into the production mode without any issues. This is usually done before the product goes into the production phase. Testing is usually of two types alpha testing and beta testing. In alpha testing, the testing is performed by the developers of the product before it is shipped to the end users for further testing and production stage. In beta testing, the end users have the hands on experience with the developed product to be tested and make a note of all the bugs they are experiencing while using the product. This enables the developers to come to know about the issues and make them before it goes live in the market for everyone.

5.2 Unit Testing

This involves performing the tests for each and every individual component of the system. It is usually done by the developers of the product. The main testing involves determining whether the presented algorithm is operating properly by making different types of input to the scheme. This is followed by verifying whether the roles included in the algorithms are returning proper values to the other modules in the application or not.
5.3 Integration Testing

This is followed by the unit testing where initially all the modules are unit tested they are all aggregated to one single unit and integration testing is performed on the aggregated unit to verify whether it is functioning as expected or not. The main objective is to test the existing interfaces between each and every module. This is performed in parallel with the development environment.

5.4 Functional Testing

The main motive is to test the functional side of the application. This can be done by verifying the functional requirement of a part of the application to make sure the application is responding to the specifications of the application. This is traced by examining the application functionality by bringing in several different types of inputs and verifying the output for every instance.

5.5 Experimental Evaluation

The test case of this project involves extracting the possible alternative options for a given query by putting out several different questions to the system containing two entities to the scheme. In this project we evaluate the performance metrics of the application using 100 questions based on mobile phones. In one test case, we are going to evaluate this 100 questions using only bootstrapping approach first. Subsequently, in test case two we are going to test the same 100 questions using Bootstrapping and K-Nearest Neighbor approach. We then make use of adjectives such as ‘better’ to identify whether the given query is a comparative or not. In the future if there is a requirement we can add
some more patterns such as ‘prefer’ and so on. These adjectives serve as the patterns to identify whether the given query is a comparative or not. Without these, it would become difficult for any end user to figure out which is a comparative query or not.

Later we begin the query search process where we input all the possible one hundred queries to the system to check the results. During this process, all the queries that were used for the evaluation purpose consisted of exactly two comparators in the input query. So, therefore the system was evaluated using two comparators which consist of 100 queries. In this project all the 100 queries consist of all different comparators. None of the queries are repeated with the same set of comparators once again. This avoids inflated output results in the system otherwise this is going to be counted as an extra result which would give high output results resulting in highest precision, recall and F1-measure. This should be avoided.

We can even test this approach using only one comparator, but since it would not make much sense for the end user as a comparative query we haven’t performed and did not include it in our research work. Here, we had begun the approach by using the pattern “which is better $C$ or $C$?” where both $C$ refer to the comparators in the question. For instance, if we include a query as “Which is better iphone 4 or galaxy grand”, iphone 4 and galaxy grand are considered to be comparators.

For questions processing, we made use of the parser chunking API which would initially scan the query to identify if it is a comparative or non comparative. The results forecasted over here are achieved by using a single pattern which is applied constantly for all the queries. Once the experiment has begun, the results are retrieved for each input
query and are recorded for manual evaluation at a later point of time which would help determine whether the results obtained for those queries are right or wrong.

While conducting this experiment session, for about 7-8 questions no answer was retrieved which explains that there was no particular match found for the given input query. For instance, for the query “which is better galaxy S6 or iPhone 6” no matching results were obtained by the system. Here, both the comparators are valid and exists in the database. We need to consider the fact that even though the answers were not retrieved for these 7-8 queries the input query that was given to evaluate the performance of the system were authentic queries which had valid comparators. These 7-8 queries are considered to be of a special case scenario as no matching answer is retrieved and are a part of the 100 queries that we had tested.

Later, we had evaluated the system with another set of questions which are considered to be invalid queries which include wrong patterns and irrelevant comparators. Since, we already know that these are invalid these are not going to be a part of 100 questions. These are not taken into consideration in evaluating the performance measurements of the system which include precision, recall and f1 measure. Hence, this is done as a part of the testing case’s purpose to see the behavior of the system to see any chances of the system giving the right results considering these invalid queries.

Once all the results are obtained for all the input queries we then do a manual inspection to verify the authenticity of those answers to arrive at a conclusion whether they are right or wrong. We then make a record of the number of right and wrong
answers for each approach so that from this we can determine the accuracy, recall and f1 measure.

The formula for determining the performance metrics are:

Precision of the system: Number of accurate results/Total number of retrieved results [35]

Recall or sensitivity of the system: Number of accurate results/Total number of relevant results [35]

The F1-measure of the system: 2 . Precision . Recall/Precision+Recall

For instance, when the end user input a query the results obtained is about 10 which is the total number of results retrieved for that query. Of those 10 results only 7 were detected to right (relevant results) and it failed to return another 5 expected or relevant results. Therefore, the total number of relevant results is 7+5 = 12. So, the precision is 7/10=0.7 in percentage it is 70% and similarly the recall calculated as 7/12 =0.5833 =58.33%. F1-measure equals 2*0.7*0.5833/0.7+0.5833=0.6363 and in percentage it is 63.63%.

This entire process is repeated for each and every individual query and their corresponding precision, recall and f1-measure values are retrieved and at the end the average for all these are figured out for the final precision, recall and f1-measure of the system. This can be done as follows:

Average (Precision) = P1+P2+…....+Pn/Total number of queries

Average (Recall) =R1+R2+……+Rn/Total number of queries.
Average (F1 measure) = F1 + F2 + …… + Fn / Total number of queries.

The right and the wrong answers for the existing (bootstrapping) and the proposed system are depicted as below in Table 5.1

**Table 5.1 Results Table**

<table>
<thead>
<tr>
<th>System</th>
<th>Existing System</th>
<th>Proposed System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Right Answers</td>
<td>80</td>
<td>92</td>
</tr>
<tr>
<td>Wrong Answers</td>
<td>20</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 5.1 depicts the entire number of correct and the wrong answers retrieved from evaluating the both the existing and proposed schemes with the same bit of queries. With the existing system, the number the percentage of correct responses to the total number of questions is 80 whereas the percentage of the proposed system is 92. The overall gain or the improvement in the system is +12%, which is a significant increase compared to the existing system.

The precision, recall and F1-Measure for the existing and the proposed system are depicted as below in Table 5.2
Table 5.2 Performance Measurements

<table>
<thead>
<tr>
<th>Methods</th>
<th>Existing System</th>
<th>Proposed System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>70%</td>
<td>87%</td>
</tr>
<tr>
<td>Recall</td>
<td>62.5%</td>
<td>75%</td>
</tr>
<tr>
<td>F1-Measure</td>
<td>66.3%</td>
<td>80.5%</td>
</tr>
</tbody>
</table>

Table 5.2 proposed system achieves higher precision, accuracy and F1-Measure when compared to the existing approach. The overall gain in the F1-Measure with the hybrid approach is 14.2%, which is a significant improvement over the existing approach.

Table 5.3 shows the results retrieved by the existing system, proposed system and Google search in relation to the query “Which is better Porsche or Lumia”.
<table>
<thead>
<tr>
<th>Existing System</th>
<th>Proposed System</th>
<th>Google Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blackberry Leap</td>
<td>Blackberry Porsche</td>
<td>Porsche P9982</td>
</tr>
<tr>
<td>Blackberry Classic</td>
<td>Nokia X</td>
<td>Porsche Hard Back Case Cover</td>
</tr>
<tr>
<td>Blackberry Passport</td>
<td>Google Nexus</td>
<td>Porsche live wallpaper</td>
</tr>
<tr>
<td>Blackberry Bold</td>
<td>Samsung Galaxy core prime</td>
<td>Nokia Lumia 530</td>
</tr>
<tr>
<td>Blackberry Bold Two</td>
<td>Samsung Galaxy grand</td>
<td>Nokia Lumia 635</td>
</tr>
<tr>
<td>Blackberry Curve</td>
<td>Samsung Galaxy S6</td>
<td>Nokia Lumia 735</td>
</tr>
<tr>
<td>Nokia 520</td>
<td>Samsung Galaxy Note</td>
<td>Nokia Lumia 930</td>
</tr>
<tr>
<td>Nokia XPlus</td>
<td>Samsung Neo</td>
<td>Nokia Lumia 1520</td>
</tr>
<tr>
<td>Nokia XpressMusic</td>
<td>HTC Desire</td>
<td>Porsche Cayman GTS</td>
</tr>
<tr>
<td>Nokia Asha</td>
<td>HTC One</td>
<td></td>
</tr>
<tr>
<td>Nokia X</td>
<td>HTC M8</td>
<td></td>
</tr>
<tr>
<td>HTC Desire</td>
<td>Sony Xperia Ultra</td>
<td></td>
</tr>
<tr>
<td>HTC One</td>
<td>Sony Xperia V</td>
<td></td>
</tr>
<tr>
<td>Device</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>Sony Xperia M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Micromax Canvas</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Karbonn Turbo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LG GFlex</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LG G2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blackberry Leap</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blackberry Classic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blackberry Passport</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apple iPhone 6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Motorola Moto X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Motorola Droid</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Motorola Droid HD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yu Yureka</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nokia XPlus</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.3 shows the differences in the results returned by the existing system, proposed system and google search. Initially, the existing system retrieves a list of
relevant results for the given query. Of those results only a very few are right answers rest others are wrong answers. When it comes to the proposed system where the search results in a list of results of which most of them are the right answers. When it comes to the Google search the approach is quite different rather than extracting the results based on one domain mobile phones it tries to retrieve a mixed set of results from different domains which include cars, mobile accessories and few relevant results of which few are the right answers.

5.6 Test Cases

The system is evaluated for testing purposes by performing certain test cases that include valid and the invalid inputs.

Test Case 1: Login

The administrator should be able to login to the system, to upload the necessary data for the end users. Similarly, the admin should be able to create a login id and password for the end users who are willing to access the system. Figure 5.1 and 5.2 show that the admin is successful in accessing the system.
Fig 5.1 Login
Test Case 2: Invalid Input Query

When the end user specifies an invalid query to the system the system scans the question and if it’s not a valid comparative question, of course it has an error allowing the end user know that the input was invalid. Figure 5.3 and 5.4 depict the error result when the user tries to input an invalid query to the system.
Fig 5.3 Invalid Input Query
Test Case 3: Valid Matching not Found

When no answer matches for the end user query, although it is a valid query the system will not pop up any results from the database for the end-user. Figure 5.5 and 5.6 clearly depict the scenario.
Fig 5.5 Input Query with no Matching
Test Case 4: Relevant Answer for a given Query

When the end user inputs a valid query to the system, the system initiates the process and retrieves the relevant result for the input query to that end user. Figure 5.7 and 5.8 depict the relevant answer for a valid query.
Fig 5.7 End user Input Query
**Fig 5.8 Relevant Results Retrieved**

<table>
<thead>
<tr>
<th>Brand Name</th>
<th>Apple</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>iPhone 6</td>
</tr>
<tr>
<td>Price</td>
<td>200.0</td>
</tr>
<tr>
<td>Screen size</td>
<td>4.7</td>
</tr>
<tr>
<td>Processor</td>
<td>2.0</td>
</tr>
<tr>
<td>Ram</td>
<td>1.0</td>
</tr>
<tr>
<td>Internal Memory</td>
<td>32.0</td>
</tr>
<tr>
<td>External Memory</td>
<td>0.0</td>
</tr>
<tr>
<td>Hits</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Brand Name</th>
<th>BlackBerry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>Passport</td>
</tr>
<tr>
<td>Price</td>
<td>245.0</td>
</tr>
<tr>
<td>Screen size</td>
<td>4.0</td>
</tr>
<tr>
<td>Processor</td>
<td>2.0</td>
</tr>
</tbody>
</table>
6. CONCLUSION AND FUTURE WORK

In this project, we implement a hybrid approach to distinguish whether a question made by the user is comparative or not and then furnish the user with possible alternative options at the same time. This Hybrid approach formed by aggregating bootstrapping and relational keyword query questioning process for text search techniques has achieved very high recall maintaining high precision. The K-Nearest Neighbor approach [34] used in relational keyword query processing is going to be very efficient in bettering the overall execution speed and equally well as achieving very high F1-Measure value. Table 5.2 depicts the evaluated proposed system performance measurements which include precision 87%, recall 75% and F1-Measure 80.5%. These results show that there is an overall gain when compared to the existing bootstrapping approach. From the above results, this system can be adapted to e-commerce and several entity recommendation systems which includes eBay, Amazon, Overstock and so on, which benefits the users providing the other options before they make any purchase as well as companies in understanding the user requirements so that they may facilitate their system as per as the user needs.

In the future, we can continue this approach for training datasets that still improve the overall functioning of the data retrieval mechanism utilizing very less time to product accurate result. In addition to this, we can make the application understand how to mine aliases and ambiguous entities for a given query [1].
BIBLIOGRAPHY AND REFERENCES

[1] Shasha Li; Chin-Yew Lin; Young-In Song; Zhoujun Li, "Comparable Entity Mining from Comparative Questions," *Knowledge and Data Engineering, IEEE Transactions on*, vol.25, no.7, pp.1498,1509, July 2013


http://dx.doi.org/10.15680/IJIRCCE.2014.0212018


http://wisdm.cs.uiuc.edu/


[34] http://www.saedsayad.com/k_nearest_neighbors.htm