Design of Query Suggestion System using Search Logs and Query Semantics

Mayank Arora¹ and Neelam Duhan²

¹M.Tech Scholar, Computer Engineering Department, YMCAUST
²Associate Professor, Computer Engineering Department, YMCAUST

Abstract

Query suggestion is an assistive technology mechanism commonly used in search engines to enable a user to formulate their search queries. Query suggestion extends the feature of search engine which helps users to describe their information needs more clearly. Building an efficient query suggestion system, however, is very difficult due to the fundamental challenge of predicting users’ search intent, especially given the limited user context information. The proposed system based on both users’ query actual semantic meanings from online dictionary sites and learning from query logs which predicts user information needs. To achieve this, the first method uses the online dictionary resource to find the context of the query and second method mines the search logs using a similarity function to perform query clustering from search logs and provides the recommendations.

Keywords: Information Retrieval, Search Engine, Query suggestion, Query Log, Semantic Search.

1. INTRODUCTION

With the growing of information on World Wide Web [1], Search Engines have become the main point of reference to access information on the web. A common fact in Web Search is that a user often needs multiple iterations of query refinement to find the desired results from a search engine. Also, Search queries are often extremely concise generally 2-3 words on average, and therefore do not adequately and/or distinctively convey users’ search intent to the search engine. This causes problem to the user to get the desired results. The user will navigate among the various URLs to get the desired results. In order to solve this problem, search engine uses another feature on the interface called Query Suggestion. Query Suggestion is thus a promising direction for improving the usability of Web search engines. This helps the search engine to know the users’ search intent. In this work, a novel design of query suggestion system for search engines is being introduced to sort out the above said problem and improve the users’ search experience.

The proposed query suggestion system tries to enhance the efficiency of the existing search engines by better formulating the user queries. It suggests K related queries where K is some user defined integer value, along with the list of URL’s with respect to the current query. When a user submits a query, the query is processed and two functions are carried out to generate query suggestions, i.e., extracting the related queries from search log and finding contextual meaning of user-query.

In the first function, the information is extracted from the search logs. The search logs contain the history of the user behaviors while working on the search engine. The extracted information is represented in the form of bipartite graph by an intermediate module. The relationship between the query and URL is drawn in the bi-partite on the basis of number of clicks on URLs with respect to the query. Afterwards, the clustering of queries is done by calculating the similarity among the various queries. It is done by calculating the normalization factor of the query using a similarity function.

In the second function, the query is analyzed with some specific criteria to determine the type of the query. The query is classified and represented in respective model to make the query more valuable. Contextual meaning of the modeled queries is found out from the online dictionary based site. The dictionary site is a powerful tool used to determine the semantics and meanings related to the query. The related meanings of the queries are found from the page source. The common and significant meanings of various queries are extracted from the result by applying ranking algorithm. The resultant terms give additional results in the form of suggestions.

The top results from both the functions are selected by the recommendation module. It suggests the required K results to the search engine interface. The search engine interface displays the query suggestion along with the top ranked URLs selected by search engine with respect to the query. The user either clicks the suggested queries, if user wants to refine his query or he will click the URL from the displayed URLs, if user finds the relevant URL. The user behaviors on the search engine are recorded in the search log to populate it with current activities which helps to refine the result in future.

The paper has been organized as follows: Section 2 describes the related work done under query suggestion system. Section 3 explains the proposed system architecture with detailed description of various functional modules. The result analysis of the work is given in Section 4 and Section 5 concludes the paper with some discussion.
2. RELATED WORK

This section describes the basic background concepts pertaining to the proposed work. A great challenge for search engines is to understand users' search intents behind queries. Query Suggestion thus, is the most challenging feature of search engine. Traditional approaches focus on the search intent related to the query but sometimes fails to provide users’ search intent related to the behavior of the user. Some Query Suggestion System uses the keyword based suggestion [5] technique which uses keywords from query traces. Interactive Query Expansion (IQE) [6] offers the users a list of suggested terms from which user can select a few augment to the original query. Some uses implicit/explicit feedbacks (e.g., [7, 8]), thesaurus (e.g., [4]), anchor texts (e.g., [9]).

Some focuses on query logs by identifying the past similar queries related to current user query. Barouni-Ebrahim and Ghorbani [11] utilize phrases frequently occurring in queries submitted by past users as suggestions. Baeza-Yates et al. [10] cluster queries present in query logs and given an initial query, similar queries from its cluster are identified based on vector similarity metrics and are then suggested to the user. In addition to user queries, query logs also contain valuable click through data and session information. Many of the previous works on query suggestion exploit this additional information. By utilizing combine data and session information, Cao et al. proposed a context aware query suggestion approach [12]. In order to deal with the data sparseness problem, they use concept based query suggestions where a concept is defined as a set of similar queries mined from the query-URL bi-partite graph. The user’s context is defined as the sequence of concepts about which the user has issued queries before submitting the current query. Given this context information, queries that are asked frequently in a given context are mined from search logs and are suggested to the user. Cucerzan and White [13] utilize user landing pages where users finally end up after post-query navigation to generate query suggestions. For each landing page of a user submitted query, they identify queries from query logs that have these landing pages as one of their top ten results. These queries are then used as suggestions. Mei et al. [14] proposed an algorithm based on hitting time on the Query-URL bipartite graph derived from search logs. Starting from a given initial query, a sub graph is extracted from the Query-URL bipartite using depth first search. A random walk is then conducted on this sub graph and hitting time is computed for all the query nodes. Queries with the smallest hitting time are then used as suggestions. Neelam Duhan, A. K. Sharma [2] pre-mines the query logs to retrieve the potential clusters of queries and then finds the most popular queries in each cluster. Each cluster entries are mined to extract sequential patterns of pages accessed by the users. The outputs of mining processes are utilized to return relevant results with popular historical queries. Yang Song et al. [15] proposed to suggest queries using Term-transition graphs in which a large amount of user preference data is mined from query logs. It constructs a term-preference graph where each node is a term in the query and each directed edge a preference. Then a topic-biased PageRank model is trained for each of the query and it guides the decision of expanding relevant terms to the original query, removing terms from the original query, or replacing existing terms with relevant terms. This model leverages a set of rich features, which includes N-gram features, external knowledge (e.g., Wikipedia, ODP) as well as the topic-biased PageRank features learnt in the previous model. Instead, the training labels were automatically inferred from the logs by leveraging a sophisticated probabilistic preference model. A critical look at the above literature indicates that no technique is semantically 100% accurate in the sense that understands human interest through contextual meaning. Also, the standard query clustering techniques have problems like choice of parameter k inputted from the user initially, selection of a center point. Most algorithms do not support the dynamic updation of clusters when new queries are available in query logs. Some of the existing recommendation systems generally recommend related queries on the basis of keyword similarity. No technique performs query recommendations based upon both clicked URLs and semantic information.

3. PROPOSED SYSTEM

Query Suggestion is a promising direction for improving the usability of Web search engines. When a user writes the query in the search textbox and submits it, the query is being processed wherein the following functions are carried out to generate query suggestions.

- Similarity between historical queries in search logs and user-query.
- Semantic meaning of user-query.

The proposed approach suggests a recommendation system that is based on the assumption that the approach will provide better results than the approaches that consider only keywords similarity to recommend related queries. Since result improvement is based on the analysis of query logs which tries to expand the queries using the keywords related to the user query and cluster formed using query logs. In addition, the results are also optimized by taking care of its semantic behavior.

3.1 Proposed System Architecture
The architecture of the proposed Query Suggestion model is shown in Figure 1. It consists of the various functional components are described in detail as given below.

![Proposed Architecture of Query Suggestion](image)

**3.1.1 Query Analyzer**

When user submits a query, the task of this module is to analyze the query. It examines the query to decide whether the query is topical or non-topical. A topical query is the query, which is generally framed by simple keywords. Assumption made by this module is that if the meaning of the query is found in the dictionary sites like bighugelabs, wordnet, then the query is topical, otherwise query is non-topical. For example “grid computing”, “data mining”, “java” etc. are topical queries, while “how to drive a car”, “program to calculate factorial” etc. are non-topical queries. After classifying the query, the control shifts to Query Modeler.

**3.1.2 Query Modeler**

In this module, query is modeled after analyzing it. If the query is non-topical, then check whether it is question based or non-question based query. If the query is question based then the question is converted into affirmative sentence by removing the starting word (W-H words). Now it becomes the non-question sentence. The stop-words like determinates from the sentence are removed by applying stop-word removal and lemmatization techniques. Prepositions tell the relationship between the adjacent words. So they are not to be removed. Then, bigram model of the user query is used to evaluate the context of the query. If the preposition is present in phrases then the phrases are converted in trigrams, otherwise it is converted in bi-gram.

For example, the submitted query is:

**Query**: - Inventory Control in Company

The resulting bigrams are:-

**N-Grams**: - Inventory Control, Control in Company

**3.1.3 Semantic Finder**

Semantics is the study of meaning or formal logics that is used for understanding human expression through language. A Semantic with learning techniques is designed to automatically extract the relevant definitions from the Online Dictionary Resource. The online resource taken is the dictionary based sites, which are a rich store of semantic definitions of terms explicitly published on the Web. For example dictionary.reference.com, www.onelook.com, www.merriam-webster.com etc. It maintains a database of definitions which referred to as Dictionary Repository, as the core of the system to accomplish its desired task. The definition here is meant to describe a semantic description of the phrases. The N-Gram phrases come from previous module are first searched in the local repository to check the presence of meaning. If the meaning is not present in the repository then it is allowed to submit the phrase on online dictionary resource. It finds the contextual meaning and related searches from the web from page source by finding some specific pattern. In this work, online resource is ask.dictionary.com which can result the definitions and related meanings. The Contextual meanings of various phrases are found out and stored in the local dictionary repository which contains the term and related meanings. The Figure 2 shows the algorithm to find the semantic meaning of user query.
The algorithm finds the semantic meaning of the queries. The query modeling module gives the set of N-Gram phrases of the query. For each phrase present in set of N-Gram phrases, search is done in the local dictionary repository. If meanings are found, then they are stored in result array, otherwise fetching of the page source is done from the dictionary based site URL by submitting the phrase. The related meanings and searches are found and stored in the result array. The result is a two dimensional array that contains the term and semantic meaning. The sample array is shown below.

$\text{Result} \begin{bmatrix} t_1 & m_1 \\ t_2 & m_2 \\ t_3 & m_3 \end{bmatrix}$

where $t_1, t_2$ are the terms and $m_1, m_2, m_3$ are the meanings.

### 3.1.3.1 Dictionary Database

The dictionary repository stores the terms and its semantics meanings in a database. This database is used by various modules to refine their search. The schema of database is shown in Figure 3.

<table>
<thead>
<tr>
<th>TERM</th>
<th>Term_ID</th>
<th>Term</th>
<th>Term_Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEMANTICS</td>
<td>Semantic_ID</td>
<td>Term_ID</td>
<td>Meaning</td>
</tr>
</tbody>
</table>

where $\text{Term\_ID}$ = unique identification number of Term Table.
$\text{Term}$ = title of the term.
$\text{Term\_Description}$ = description of the term.
$\text{Semantic\_ID}$ = unique identification number of Semantic Table.
$\text{Meaning}$ = semantic meaning of the term.

### 3.1.4 Query Semantic Ranker

The Semantic meanings of the various phrases have been found from previous modules. The useful meanings of the various phrases are to be found with respect to the query. For this purpose, Ranking of the meanings of various phrases is needed in order to retrieve the efficient suggestions. Four steps will be taken by this Ranker module.

- It finds the frequency of various results of the bigram queries as submitted to online dictionary resource.
- Remove the duplicate or ambiguous terms from the results.
- Sorts the results in descending order
- Extract the top N recommendations.
Figure 4 Algorithm for Query Semantic Ranking

An algorithm is shown in Figure 4 to rank different meanings or suggestions. It finds out the common terms in at least any of two phrases. If the number of meaningful terms obtained from bigram queries is less than required (say two), then we will give the preference to the meaning of whole query. The priority is given on the basis of frequency of terms in the phrases. The top N recommendations are passed on to the recommendation module.

3.1.5 Cluster Generator

The source of data used by this module is from Search logs, where large samples of users’ online browsing activities are stored. A search log as a stream of query and click events. A typical Search log entry contains the following fields: a unique identification number, the timestamp of the visit, type of the Event (QUERY/CLICK) and the URL that the user visited. The sample of search log is shown in Table 1.

<table>
<thead>
<tr>
<th>ID</th>
<th>TIMESTAMP</th>
<th>SESSIONID</th>
<th>EVENTTYPE</th>
<th>EVENTVALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2013051208154</td>
<td>101</td>
<td>QUERY</td>
<td>types of expert system</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2013051208064</td>
<td>101</td>
<td>CLICK</td>
<td><a href="http://www.experts">www.experts</a> system.net</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2013051210114</td>
<td>102</td>
<td>CLICK</td>
<td><a href="http://www">www</a>. website.com</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

According to the number of clicks on the URL, the click count is recorded with the query in the search engine logs. A Bi-Partite graph is made on the basis of set of unique queries and URLs. The clustering of various queries is done on the basis of similarities. The basic assumption is that the two queries are similar to each other if they share a large number of clicked URLs.

A query node is created for each unique query in the log. Similarly, a URL node is created for each unique URL in the log. An edge eij is created between query node qi and URL node uj if uj is a clicked URL of qi. The weight wij of edge eij is the total number of times when uj is a click of qi aggregated over the whole search log. A sample Bi-Partite graph is shown in Figure 5 for four queries and five clicked URLs. The count on each edge shows the number of clicks corresponding to a query.
The bipartite graph can help us to find similar queries. From the bipartite graph, we represent each query $q_i$ as a normalized value. Suppose $Q$ and $U$ be the sets of query nodes and URL nodes, respectively. The $j^{th}$ element of the feature vector of a query $q_i \in Q$ is

$$
(q_i)_j = \begin{cases} 
\frac{\text{norm}(w_j)}{Q} & \text{if edge } w_j \text{ exists} \\
0 & \text{otherwise} 
\end{cases}
$$

where

$$
\text{norm}(w_j) = \frac{\sqrt{\sum_{i=1}^{N} w_{ij}^2}}{N}
$$

The similarity value between two queries $q_i$ and $q_j$ is measured by the normalization value between their normalized feature vectors. That is,

$$
\text{Similarity}(q_i, q_j) = \frac{\text{Normalization value of common visited URLs of } q_i \text{ and } q_j}{\text{Normalization value of total visited URLs of } q_i \text{ and } q_j}
$$

where the normalization value is calculated by (1).

The threshold value of the cluster is set according to the number of queries present in search logs and number of cluster formed in clustering database. Initially, data present in the query log is very less. In order of to make the cluster with less data, the system has to make the value of threshold to be small. If the number of clusters is small then the value of the threshold is less. After populating the data in the search logs which is enough to make large number of clusters, the value of the threshold is updated accordingly.

Each cluster contains the queries which are most similar to each other. In addition, each cluster also stores the keywords of the related queries. Keyword is the individual word present in each query. The keywords of the queries present in clusters are stored in clustering database. For Example,

$$
C = \{\text{data mining, data warehousing, knowledge discovery process}\}
$$

Keyword(C) = \{\text{data, mining, data, warehousing, knowledge, discovery, process}\}

The algorithm for this module is shown in Figure 6.

### 3.1.6 Incremental Clustering

Incremental Clustering is a process to update the cluster periodically in a fixed interval of time. The periodic updating of clusters is required due to the dynamic nature of databases. The effective use of background knowledge allows getting more information, to reduce blindness and improve the clustering quality.

#### 3.1.6.1 Incorporation of newly submitted queries

The clusters are formed from the search logs by using the Bi-Partite graphs. The normalization vector is calculated and stored in the clustering database. Also, the keywords in the clusters are also stored in the clustering database. These keywords are used to again cluster the new query submitted by the user. The keywords from the user query are extracted and also find the keywords in the clusters. The similarity value of the keywords is calculated from (4). Whose similarity keyword value is greatest among all values, the query goes to that cluster. If, in case, the similarity keyword value of the entire cluster with the query is 0, then new cluster is formed and updated in the database.

#### 3.1.6.2 Updatation of existing cluster

The existing cluster is updated time to time to make cluster stricter. In this process, we find out the similarities among the queries in each cluster. The normalization value of the queries with respect to the URLs is found out which tells the relationship between the URL and query. The similarities of the queries are again determined from the new value. The main aim of incremental clustering is to use the new similarity threshold value from the previous value for clustering in order to achieve the efficient result for suggestion. The similarity threshold value is set on the basis of the data present.
in the search logs. Heuristic techniques typically determine cluster, based directly on the previous user activities. On the basis of similarity threshold, the cluster is updated periodically.

**Algorithm: Query_Clustering( Q[], U[], T )**

**Input:** the set of queries Q, set of URLs U and the threshold T;

**Output:** the set of clusters C;

Begin  
  k = 1;  //k is the number of clusters  
  for (i = 1; i <= Length (Q); i ++) do  
  {  
    Set ClusterId (Q[i]) = Null;  //initially no query is clustered  
  for (j = 1; j <= Length (U); j ++) do  
  {  
      Calculate normalization vector q^j from (1)  
  }  
  }  
  for (each query A ∈ Q) do  
  {  
    ClusterId (Q[i]) = Ck  
    Ck = {A}  
    for (each query B ∈ Q such that A ≠ B) do  
    {  
      Calculate the Similarity (A, B) using (3) for every URLs U  
      if (Similarity (A, B) >= T) then  
      {  
        Set ClusterId(B) = Ck;  
        Ck = Ck U {B};  
      }  
      else  
      Continue;  
    }  
    k = k+1;  
  }  
  Return the cluster set C  
End

**Figure 6 Algorithm for Query Clustering**

### 3.1.7 Similarity Analyzer

Similarity Analyzer analyzes the user query submitted on search engine with the existing cluster database. In Similarity Analyzer module, the similarities between the user query and clusters are determined. The keywords from the user query are extracted on some specific pattern. Keywords are the individual term present in each query after stemming and stop word removal techniques. Also, the queries present in the clusters are determined. The individual terms are extracted from each query. These terms of the query present in respective cluster becomes the keywords of the cluster. These keywords are stored in the clustering database. Then, the similarity between the user query and cluster is determined on the basis of its keywords. The similarity of keywords is found using keyword similarity function as given below.

\[
\text{Similary((A,B))} = \frac{|\text{Keyword}(A) \cap \text{Keyword}(B)|}{|\text{Keyword}(A) \cup \text{Keyword}(B)|}
\]

where Keyword (A) and Keyword (B) are the sets of Keywords in the queries A and B respectively.

The similarity keyword function computes some value. The largest value of similarity keyword function is found out. The cluster having largest value is selected. For example, suppose the user query, Q is “history of apple”. The keywords of user query is KW(Q)= {history, apple}. Let us assume we have two clusters C1 and C2 respectively. The query in cluster C1= {apple iTunes, Steve jobs, Introduction to apple, history of apple company} and cluster C2= {apple iPods, iPhone, apple computer, MAC, iPads}. The Keyword (C1) = {apple, iTunes, Steve, jobs, Introduction, apple, history, apple, company}. The Keyword (C2) = {apple, iPods, iPhone, apple, computer, MAC, iPads}.

The Similarity keyword function is calculated by:

\[
\text{Similary((Q, C1))} = \frac{|\text{Keyword}(Q) \cap \text{Keyword}(C1)|}{|\text{Keyword}(Q) \cup \text{Keyword}(C1)|} = 0.44
\]
The relevancy of suggestion is based on historical behavior of other users registered in search logs. The primary priority is given to the results of query cluster ranking technique because the recommendation system recommends related suggestion corresponding to a user query and top L recommendations are passed on to the recommendation module. The relevancy of suggestion is based on historical behavior of other users registered in search logs.

The next section combines the results from both the techniques to generate query suggestions.

### 3.1.8 Query Clustering Ranker

The cluster, whose similarity keyword value is greater, is chosen. The queries in selected cluster are determined. These are ranked by Query Cluster Ranker Module. The algorithm is based on a clustering ranking process in which groups of similar cluster with the query are identified. The similarity values of the queries are sorted in descending order. Ranking is done on the selected cluster on the basis of similarity function value. The top m recommendations are passed on to the recommendation module. The relevancy of suggestion is based on historical behavior of other users registered in search logs.

### 3.1.9 Recommendation Module

The Recommendation Module recommends K related suggestion corresponding to a user query and top L recommended URLs to the query submitted by the user to search engine. Ranking and returning the most relevant results of a query is a popular paradigm in Information Retrieval. The query recommendation system recommends related suggestion by combining the two methods. The first method to be used is the Semantic Search recommendation with the help of dictionary related sites after modeling the query and ranking module selects top n suggestion. A second method is the query suggestion by using the search logs. The queries are clustered into groups and find the similarity with the user query. The recommendation module selects top \((m + n)\) suggestions where \((m + n)\) is less than or equal to 8 (in the current scenario). The primary priority is given to the results of query cluster ranking technique because the main user behavior is present in search logs. The semantic search determines the general meaning. The user interest is present in search logs.

### 4. RESULT ANALYSIS

We compared the results of our proposed work with the Google query suggestion system with the help of graph. For the comparison purposes, two metrics are used to measure the performance, i.e., Precision Factor and Average Relevance Factor. Precision is defined as a metric to ensure that the query returns all related suggestions. In other words, Precision is the fraction of number of relevant suggestions to the total number suggestions returned by the system. The Average Relevance Factor depicts the average precision of the proposed suggestion system corresponding to a set of queries submitted by the user.

\[
\% \text{ Precision} = \frac{\text{Number of relevant Suggestions}}{\text{Total Number of Suggestions}} \times 100
\]  

\[
\% \text{ Average Relevancy} = \frac{\text{Examples of user’s Reviewed Relevant Suggestion per Q}}{Q} \times 100
\]

where \(N\) is the total number of queries and \(Q\) is the set of queries.

### 4.1. Result Analysis with Examples

We conducted a survey of 20 users and take the feedbacks in which we take four queries for which comparison is done between the Google suggestion and our proposed work as shown in Table 2.

<table>
<thead>
<tr>
<th>Query</th>
<th>Google</th>
<th>Proposed Work</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple</td>
<td>Apple fruit</td>
<td>Apple Store Online</td>
</tr>
<tr>
<td></td>
<td>Apple iPhone</td>
<td>Apple IPhone</td>
</tr>
<tr>
<td></td>
<td>Apple iPad</td>
<td>Apple Official Website</td>
</tr>
<tr>
<td></td>
<td>Apple Store</td>
<td>Apple IPod</td>
</tr>
<tr>
<td></td>
<td>Apple ITunes</td>
<td>Apple the Fruit</td>
</tr>
<tr>
<td></td>
<td>Apple TV</td>
<td>Apple Stores</td>
</tr>
<tr>
<td></td>
<td>Apple daily</td>
<td>Apple Logo</td>
</tr>
<tr>
<td></td>
<td>Apple iPod</td>
<td>Apple IPad</td>
</tr>
<tr>
<td>Expert System</td>
<td>Types of expert system</td>
<td>Examples of expert system</td>
</tr>
<tr>
<td></td>
<td>Examples of expert system</td>
<td>Types of expert system</td>
</tr>
<tr>
<td></td>
<td>Expert system pdf</td>
<td>Characteristics of expert system</td>
</tr>
<tr>
<td></td>
<td>Expert system ppt</td>
<td>History of expert system</td>
</tr>
<tr>
<td></td>
<td>Components of expert system</td>
<td>Expert system Tutorials</td>
</tr>
<tr>
<td></td>
<td>Expert system architecture</td>
<td>Medical Expert System</td>
</tr>
<tr>
<td></td>
<td>Artificial intelligence</td>
<td>Expert System Introduction</td>
</tr>
</tbody>
</table>
According to the survey, the users get relevant results from Google and our proposed system. The Precision factor of both Google and Proposed Work is shown in Table 3.

### Table 3: Calculation of Precision factor

<table>
<thead>
<tr>
<th>Query</th>
<th>Google</th>
<th>Proposed Work</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple</td>
<td>62.5 %</td>
<td>75 %</td>
</tr>
<tr>
<td>Expert System</td>
<td>50 %</td>
<td>87.5 %</td>
</tr>
<tr>
<td>Cloud Computing</td>
<td>62.5 %</td>
<td>62.5 %</td>
</tr>
<tr>
<td>Internet Protocols</td>
<td>50 %</td>
<td>75 %</td>
</tr>
</tbody>
</table>

The Average Relevancy Factor (as given in (6)) of both Google and proposed system is:

\[
\text{Average Relevancy of Google} = \frac{0.625 + 0.5 + 0.625 + 0.625}{4} \times 100 = 56.25 \%
\]

\[
\text{Average Relevancy of proposed system} = \frac{0.75 + 0.875 + 0.625 + 0.75}{4} \times 100 = 75 \%
\]

The Graphical Representation of Precision Factor is shown in Figure 7.

![Figure 7 Graph Representation of Precision](image)

The Graphical Representation of Average Relevancy Factor is shown in Fig. 8.

![Figure 8 Graph Representation of Average Relevancy](image)
4.2. Screenshot of Implementation

The Query Suggestion System was implemented in ASP.NET 4.0 using C# as front end and MS SQL Server 2008 at the back end to support definition repository. Figure 9 shows the interface of search service.

![Figure 9 Search Interface](image1)

Figure 9 Search Interface

Figure 10 shows the search result interface after submitting a query “expert system”. It gives the suggestion at the left side of the interface along with the list of URLs.

![Figure 10 Search Result](image2)

Figure 10 Search Result

Separate software is developed for the offline step as shown in Figure 1. In this, the bipartite graph is developed from the search log. The normalization value is calculated for each edge present in the graph. Then, the query is clustered on the basis of similarity factor as shown in Figure 11. After the cluster is formed, the keywords present in the each cluster are extracted and stored in the repository.
5. CONCLUSION

A novel design of query suggestion system based on click through data in query log and semantic search has been proposed for implementing effective web search. The most important feature is that the proposed approach is based on users’ behavior, which determines the relevance between Web pages and user query words. In addition, the results are also optimized by taking care of its semantic behavior. By this way, the time user spends for seeking out the required information from search result list can be reduced and the more relevant Web pages can be presented. The results obtained from practical evaluation are quite promising in improving the effectiveness of interactive web search engines. The experimental results clearly show that our approach outperforms two baselines in both coverage and quality. It covers the semantic relation with the query and user behavior from search log. It improves quality by using different ranking techniques on the results.

References


