ABSTRACT

BACKGROUND: Nowadays most people daily activity is to search for information on the Web. OBJECTIVE: In web searching, the information retrieved is not appropriate, because it gives ambiguous information for the user query, and the user cannot get relevant information within a time. RESULTS: We proposed a new methodology for information retrieval system by providing the top most exact information to the user. This system aims to provide the correct information to the user from the document collection and communicated to the system by means of user-initiated query. CONCLUSION: The information is given to the end-user with the top most ranking data within the stipulated time and the remaining top information will be moved to the data repository for future use. In this system the proposed algorithm has produced better results than existing system and it costs less online computation time.

KEYWORDS: Search Engine, Information Retrieval, Text mining, Term Frequency.

INTRODUCTION

Web searching and navigating have become part of our daily online lives. In web search, the system has to search over billions of documents available on many computers. Search engine has become essential, and provides an endless amount of information that the web contains. Since the inception of the web, the information extraction has to deliver a content that satisfies the users with increasingly accurate results through the efficient retrieval algorithms. This system provides relevant documents from the collection, and the user interacts with the system through queries [10]. The user’s query contains a topic that conveys the information about the user desires. The document is said to be relevant if and only if it is perceives information with respect to the user requirement.

The technologies used for searching and navigation are central to the smooth operation of the web and it is hard to imagine finding information without them [1]. The usual search scenario involves someone is typing a query to a search information and receiving results with a list of ranked documents in order. The resulting web pages returned by a search engine are given as a ranked list of the web pages and it is considered as most relevant to the user query. These returned answers are not always precise, in the sense that the search engine is trying to find the “best” match for the query, and the person browsing the web pages returned must decide whether any specific page satisfies his or her query or not. The big issue is to collect documents for indexing, and the systems works efficiently for handling the exploitation of hypertext.

Some of the mining methodology provides the content to the maximum exact but not within the time and it has been accumulated by the most adherent relevant information. To overcome this issues Search information and Retrieval (SIR) has been proposed to extract data with respect to the relevant data, data generalization, data
clustering and analyse the information [8]. In this facilitation our system added the mining with the exact content to the end-user by framing the appropriate data set and ranking the data with the hierarchy. The time factors maintained to give the exact information with the top most overlooked information. Meta data only stored in the data repository and used the meta crawler to deliver mined content to the user.

In this work, we proposed a new methodology by forming the user query as a new data for the mining execution. Surfers are turning to search engines to provide them with an entry point web page to help them satisfy their information needs. In SIR system strategy, a user seeking information on the Web will normally iterate through the formulating Queries, Selection, Navigation, and modification of query. In Query formulation, the user submits a query to get the information relevant to their goal; normally a query consists of one or more input keywords [9]. Then the user selects one of the web pages from the ranked results list returned by the search engine, clicks on the link to that page, and browses the page once it is loaded into the browser. The user initiates a navigation session, which is the process of clicking on links and browsing the pages displayed is shown in Fig.1. The user surfing the Web by following links will use various cues and tools to augment his or her navigational activity. Sometimes a navigation session may be interrupted when the user decides to modify the query, and the original query is reformulated to resubmit it to the system.

Fig.1: Example of Web Navigation

Overview of the Proposed Work:

In this system begins with the data set formation from the indexing and the query process shown in Fig2. Index the documents in advance makes the system to enable searching easily, and the query process uses query to produce a ranked list of documents [9]. The indexing process creates index using the functions for acquisition of text, text transformation [4]. The text acquisition function is to identify and make available the documents from an existing collection that will be searched by the user.

Fig. 2: Architecture diagram of SIR system
The next step in indexing process, the documents are passed, and the text acquisition step creates a document with the text and meta information for all the documents and stored it in a data store [4]. The meta information is about a document that is not part of the text content. The text transformation step transforms documents into index that are the parts of a document that are stored and used in searching. The set of all the terms that are indexed for a document collection is called the index vocabulary. This output is fed into the index creation function to create indexes for fast searching. This system provides the index creation in efficient manner, depends upon the response to the query. Indexes must also be able to be efficiently updated when new documents are acquired.

The query process contains the tasks for user interaction, ranking the documents, and evaluation [6]. The user interaction provides the interface to the user to access the system. First the user’s query is converted into index terms and it takes the ranked list of documents from the system and then it is organized into the results shown to the user. The snippets are generated to summarize documents and the document is the sources of information used to give better results. The ranking process takes the transformed query from the user interaction function and calculates score to generate a ranked list of documents [7]. Since many queries may need to be processed in a short time, ranking should be both efficient, and effective. The quality of the ranking determines whether the system find the relevant information according to the user query. Finally the efficiency of the system is measured using evaluating process and it is used to record and analyze user behaviour using log data. The evaluation process results are used to tune and improve the ranking function.

**SIR system Indexing Process:**

The primary responsibility of SIR system is to identify and acquiring documents and it can efficiently handle the huge volume of new pages on the Web. It ensures that the pages may have changed since the last time a crawler visited a site are kept “fresh” for the system. The documents are converted into a consistent text plus metadata format [4]. The document data store is a database used to manage large numbers of documents and the structured data that is associated with them. The document contents are typically stored in compressed form for efficiency. In this system the metadata about the document and other related information hauled from the documents are stored in more efficient storage system to provide very fast.

![Fig.3: Process of Tokenization](image)

Text Transformation performs Parsing, stopping, stemming, and linking functions for information extraction given in Fig3. The parsing is responsible to recognize structural elements in the document by processing the sequence of text tokens such as titles, figures, links, and headings. The document and query text both should be transformed into tokens for easy comparison. The stopping has the simple task of removing common words from the stream of tokens like a, an, and, the, to etc. Removing them can reduce the size of the indexes considerably. The task of the stemming is to group words that are derived from a common stem [3] [5].For example “walk”, “walked”, and “walking”. By replacing each member of a group with one designated word we increase the likelihood that words used in queries and documents will match. Term frequency count is calculated using the TF-IDF weight. This weight is a statistical measure used to evaluate importance of a word in a document collection. The importance of the word depends on the number of times a it appears in the document. SIR system uses this weighting scheme to calculate the score and ranking a document’s that are relevance to a user query [7].

**Index Creation:**

First the SIR system gathers words, features, and documents and the ranking function uses this information for calculating scores for documents. The types of data generally required are the counts of index term occurrences in individual documents, the positions in the documents where the index terms occurred, the counts of occurrences over groups of documents and the lengths of documents in terms of the number of tokens. In this
system a document and a query both are considered as vectors corresponding to all the keywords to compute similarity between these two vectors [6]. The calculated similarity values are used for ranking the documents. Starting with a set of d documents and a set of t terms, we can model each document as a vector v in t dimensional space R [2]. The term frequency be the number of occurrence of term t in the document d, that is, freq(d,t) and the example is given in Table 1.

Table 1: Frequency of terms per document

<table>
<thead>
<tr>
<th>document/item</th>
<th>t1</th>
<th>t2</th>
<th>t3</th>
<th>t4</th>
<th>t5</th>
<th>t6</th>
<th>t7</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td>1</td>
<td>5</td>
<td>9</td>
<td>10</td>
<td>0</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>d2</td>
<td>6</td>
<td>18</td>
<td>6</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>d3</td>
<td>16</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>9</td>
<td>10</td>
<td>17</td>
</tr>
<tr>
<td>d4</td>
<td>20</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>5</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>d5</td>
<td>0</td>
<td>9</td>
<td>21</td>
<td>9</td>
<td>4</td>
<td>0</td>
<td>12</td>
</tr>
</tbody>
</table>

The term frequency value is non zero if the document contains the term, otherwise it is zero. The relative term frequency is measured using the frequency of terms and the total number of occurrence of terms in the document is given in Table 2. The term frequency is computed in Eq. (1)

$$TF(d,t) = \begin{cases} 0 & \text{if } freq(d,t) = 0 \\ 1 + \log(1 + \log(freq(d,t))) & \text{otherwise} \end{cases} \tag{1}$$

Table 2: Term Frequency (TF)

<table>
<thead>
<tr>
<th>document/item</th>
<th>t1</th>
<th>t2</th>
<th>t3</th>
<th>t4</th>
<th>t5</th>
<th>t6</th>
<th>t7</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td>1</td>
<td>1.23</td>
<td>1.29</td>
<td>1.3</td>
<td>0</td>
<td>1.26</td>
<td>0</td>
</tr>
<tr>
<td>d2</td>
<td>1.25</td>
<td>1.35</td>
<td>1.24</td>
<td>1.3</td>
<td>0</td>
<td>1</td>
<td>1.3</td>
</tr>
<tr>
<td>d3</td>
<td>1.343</td>
<td>0</td>
<td>1.3</td>
<td>0</td>
<td>1.29</td>
<td>1.3</td>
<td>1.34</td>
</tr>
<tr>
<td>d4</td>
<td>1.36</td>
<td>0</td>
<td>1.11</td>
<td>0</td>
<td>1.23</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>d5</td>
<td>0</td>
<td>1.29</td>
<td>1.36</td>
<td>1.29</td>
<td>0</td>
<td>1.2</td>
<td>1.317</td>
</tr>
</tbody>
</table>

The inverse document frequency (IDF) is another measure that represents the importance of a term t. If a term t occurs in many documents, its importance will be reduced. The formula for IDF is given in Eq.(2) and example is given in Table 3.

$$IDF(t) = \log \frac{1 + |d|}{|df|} \tag{2}$$

Table 3: Inverse Document Frequency (IDF)

<table>
<thead>
<tr>
<th>document/item</th>
<th>t1</th>
<th>t2</th>
<th>t3</th>
<th>t4</th>
<th>t5</th>
<th>t6</th>
<th>t7</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDF(t)</td>
<td>0.18</td>
<td>0.3</td>
<td>0.08</td>
<td>0.3</td>
<td>0.48</td>
<td>0.18</td>
<td>0.3</td>
</tr>
</tbody>
</table>

An Index term weight is used to find importance of terms in documents and calculates scores for ranking. The weighting function calculates weights using the document statistics and stores them in lookup tables. Weights could be calculated as part of the query process, and some types of weights require information about the query, and the frequent calculation during the indexing process will improve the efficiency of the query process. This model uses as TF-IDF weighting and it is based on a combination of the frequency or count of index term occurrences in a document. In this model, the TF-IDF weight is calculated by multiplying these values shown in Eq.(3).

$$TF - IDF(d,t) = TF(d,t) \times IDF(t) \tag{3}$$

The inversion function is used to change the stream of document-term information for the creation of inverted indexes. The document indexes are distributed for indexing process and the query process [9]. The basic idea is representing a document and a query both as vectors and uses similarity measure for ranking documents. In information retrieval system the first step is to identify keywords for representing the documents, and indexing useless words are avoided, a text retrieval system often associates stop list with a set of documents [5]. The irrelevant words are called stop list such as the, of, for, with and so on.

The information retrieval system needs to identify groups of words where in a group are small syntactic variants of one another and collect only the common word stem per group. The common word stem may be shared among the different words such as apple, apples. The available set of d documents and a set of t term, and the term frequency is calculated using the number of occurrence of term t in the document d, that is, freq(d,t). The term frequency matrix TF(d,t) measures the associated of a term t with respect to the given document d. It is defined as 0 if the document does not contain the term and non zero otherwise. The relative
term frequency is measured using the term frequency versus the total number of occurrence of all the terms in the document given in Eq.(1). There is another important measure given in Eq.(2), called inverse document frequency (IDF), that represents the importance of a term t in the documents. The importance of term will be reduced when it occurs in many documents. Consider there are seven terms and five documents. The above mentioned Equations are applied and the calculation is given below. Table 1 shows a term frequency matrix where each row represents document, each column represents a term, and each entry registers freq(di, tj), the number of occurrence of term tj in document di. The tf–idf weight of a term is the product of its weight is shown in Eq.(4) and calculated values are given in Table 4.

\[ W_{t,d} = tf_{t,d} \times \log \frac{N}{df_t}, \]  

(4)

**Table 4: Calculation of weight**

<table>
<thead>
<tr>
<th>document /term</th>
<th>t1</th>
<th>t2</th>
<th>t3</th>
<th>t4</th>
<th>t5</th>
<th>t6</th>
<th>t7</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td>0.18</td>
<td>0.37</td>
<td>0.1</td>
<td>0.39</td>
<td>0</td>
<td>0.22</td>
<td>0</td>
</tr>
<tr>
<td>d2</td>
<td>0.22</td>
<td>0.41</td>
<td>0.1</td>
<td>0.39</td>
<td>0</td>
<td>0</td>
<td>0.39</td>
</tr>
<tr>
<td>d3</td>
<td>0.24</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0.62</td>
<td>0.23</td>
<td>0.4</td>
</tr>
<tr>
<td>d4</td>
<td>0.24</td>
<td>0</td>
<td>0.09</td>
<td>0</td>
<td>0.59</td>
<td>0.18</td>
<td>0</td>
</tr>
<tr>
<td>d5</td>
<td>0</td>
<td>0.39</td>
<td>0.11</td>
<td>0.39</td>
<td>0</td>
<td>0.21</td>
<td>0.4</td>
</tr>
</tbody>
</table>

The document collection is static if the collections are changed infrequently. But mostly the collections are dynamic because it is modified frequently. This adds overhead for adding the new terms into the dictionary. The reconstruction of index from the beginning is explained in the above mentioned algorithm. If there is a requirement of adding the new documents, one solution is to maintain two indexes: permanent index and a temporary index that stores new documents. Deletions are stored using invalidation bit and deleted documents are filtered before returning the search result. Each time the temporary index becomes too large, and then it is merged into the permanent index.

**SIR system query Handling:**

In this section, we will focus on the features of the queries and documents that are most important for the ranking algorithm. The query input function provides an interface and a parser for a query language. A typical web query consists of a small number of keywords with no operators. A keyword is simply a word that is important for specifying the topic of a query [9]. The query transformation includes a range of techniques that are designed to improve the initial query, both before and after producing a document ranking. Tokenizing, stopping, and stemming must be done on the query to produce index terms that are comparable to the document terms [2]. Spell checking and suggesting queries to user are query transformation techniques that produce similar output. In both cases, the user is presented with alternatives to the initial query that are likely to either correct spelling errors or be more specific descriptions of their information needs. These techniques often leverage the extensive query logs collected for web applications. Query expansion techniques also suggest or add additional terms to the query, but usually based on an analysis of term occurrences in documents [2]. This analysis may use different sources of information, such as the whole document collection, the retrieved documents, or documents on the user’s computer. Relevance feedback technique expands queries based on term occurrences in documents that are identified as relevant by the user.

**Functionality Similarity Calculation:**

Jaccard similarity coefficient (JSC) is used to compute functionality similarity and it is the statistical measure for calculating similarity between samples sets [7]. The JSC is defined as a cardinality of intersection is divided by the cardinality of their union for both sets. Concretely the formula for computing similarity between a and b is given in Eq 5.

\[ D_{sim}(a, b) = \frac{|D_a \cap D_b|}{|D_a \cup D_b|} \]  

(5)

This can be inferred from this formula that the larger \( D_a \cap D_b \) is, the more similar the two services are. From the above Division \( D_a \cup D_b \) is the scaling factor that ensures that description similarity ranges from 0 to 1. Similarly the Functionality similarity is calculated as given in Eq 6.

\[ F_{sim}(a, b) = \frac{|F_a \cap F_b|}{|F_a \cup F_b|} \]  

(6)

The weighted sum of description similarity and functionality similarity is used to compute characteristic similarity between a and b is given in Eq 7.
\[ C_{\text{sim}}(a, b) = \alpha \times D_{\text{sim}}(a, b) + \beta \times F_{\text{sim}}(a, b) \] (7)

In this formula, \( \alpha \in 0, 1 \) is the description similarity weight and \( \beta \in 0, 1 \) is the weight of functionality similarity. The relative importance between these two expressed using weight. In the recommender system, for the total \( n \) services provided, calculate the characteristic similarities of every pair of services and \( n \times n \) characteristic similarity matrix \( M \) is formed. An entry \( m_{ab} \) in \( M \) represents the characteristic similarity between \( a \) and \( b \).

The scoring function calculates scores for documents using the ranking algorithm. The input to the ranking is a training set consisting of partial rank information for a set of queries \((q_1, r_1), (q_2, r_2), \ldots, (q_n, r_n)\) where \( q_i \) is a query and \( r_i \) is partial information about the desired ranking, or relevance level, of documents for that query. This means that if document \( A \) should be ranked higher than \( B \), then \((A, B) \in r_i\); otherwise, \((A, B) \in \emptyset \). For example, if a person clicks on the third document in a ranking for a query and not on the first two, we can assume that it should be ranked higher in \( r \). If \( d_1, d_2, \) and \( d_3 \) are the documents in the first, second, and third rank of the search output, the data will result in pairs \((d_3, d_1)\) and \((d_3, d_2)\) being in the desired ranking for this query.

Let’s assume that we are learning a linear ranking function \( w \cdot d_a \), where \( w \) is a weight vector that is adjusted by learning, and \( d_a \) is the vector representation of the features of document \( A \). These features are, based on page content, page metadata, anchor text, links, and user behavior. Instead of language model probabilities, however, the features used in this model depends on the match between the query and the document. For example, there may be a feature for the number of words in common between the query and the document body, and similar features for the title, header, and anchor text. The weights in the \( w \) vector determine the relative importance of these features. If a document is represented by three features with integer values \( d = (3, 2, 1) \) and the weights \( w = (1, 2, 1) \), then the score computed by the ranking function is just:

\[ w \cdot d = (3, 2, 1) \cdot (1, 2, 1) = 3 \cdot 1 + 2 \cdot 2 + 1 \cdot 1 = 8 \]

This simply means that for all document pairs in the rank data, and the score for the document with the higher relevance rank to be greater than the score for the document with the lower relevance ranking. The document scores must be calculated for all document pairs in the rank data, and the score for the document with the higher relevance rank to be greater than the score for the document with the lower relevance ranking. The ranking function is free from performance-enhancing changes. Realistically, however, the priority queue \( R \) only needs to hold the top \( k \) results at any one time. If the priority queue ever contains more than \( k \) results, the lowest-scoring documents can be removed until only \( k \) remain, in order to save memory. Also, looping over all documents in the collection is unnecessary; so the algorithm changes the score of the documents in the inverted lists. The only major use of memory comes from the priority queue, which only needs to store \( k \) entries at a time. However, in a realistic implementation, large portions of the inverted lists would also be buffered in memory during evaluation.

**Efficiency of the Proposed System:**

A hybrid system will make use of the combination of collaborative content based filter and it restricts itself with the collaborative filtering strategies. Before the content based filtering begins it accepts dataset formed in the process and with the hierarchy of the user preference the recommended data set are formulated and the ranking will be given based on the user preferences [7]. In a content-based system, keywords are used to describe the items; besides the user profile is built to indicate the keyword item to which the user specified their desires[7]. In other words, the algorithm proposed try to give the most relevant data in the hierarchy for the recommended user and also it make sure that there is no similarity in the user retrieved information and further unused and used information will be stored in the repository for future use [7]. Generally the cold start problem, data sparsity may affect the system performance and here we discussed how the proposed system overcomes these problems to improve the performance of the system.

SIR system uses two most common measures recall and precision, to summarize and compare search results. The recall is used to measure how much of relevant documents that are retrieved for a given query, and precision measures about rejecting the number of documents which are not relevant to user query. In this, A is
the relevant set of documents for the query, and \( B \) is the set of retrieved documents. Then these measures can be calculated using the Eq. (8) and (9).

\[
\text{Recall} = \frac{|A \cap B|}{|A|} \quad (8)
\]

\[
\text{Precision} = \frac{|A \cap B|}{|B|} \quad (9)
\]

The precision and recall together characterize the effectiveness of a search as a classifier and the precision is more meaningful to the user of a system. If a system retrieved 20 documents for a given query, and 15 documents are relevant then the precision value is of 0.75. This system is trained to minimize classification errors.

The documents that are retrieved must be defined based on the ranking to measure recall and precision. The top ten documents of two possible rankings, together with the recall and precision values calculated at every rank position for a query that has six relevant documents. These rankings might correspond to, the output of different retrieval algorithms. When ten documents are retrieved at rank position 10, the two rankings have the same effectiveness as measured by recall and precision. The recall and precision for ranking is shown in Fig 4 and Fig 5.

![Fig.4: Recall and precision values for ranking 1](image1)

![Fig.5: Recall and precision values for ranking 2](image2)

In the following discussion of averaging techniques, the two rankings shown in Fig.6 and 7 are used as a running example. These rankings come from using the same ranking algorithm on two different queries. Then the average is used to summarize the effectiveness of a specific ranking algorithm across a collection of queries. Different queries will often have different numbers of relevant documents, as is the case in this example, the recall precision values calculated for the top 10 rank positions. Given that the average precision provides a number for each ranking, the simplest way to summarize the effectiveness of rankings from multiple queries would be to average these numbers. This effectiveness measure, mean average precision, or MAP, is used in most research papers and some system evaluations. In some evaluations the geometric mean of the average precision (GMAP) is used instead of the arithmetic mean. This measure, because it multiplies average precision values, emphasizes the impact of queries with low performance. It is defined in Eq. (10)

\[
GMAP = \exp \left( -\frac{1}{n} \sum_{i=1}^{n} \log \text{AP}_i \right) \quad (10)
\]

Where \( n \) is the number of queries, and \( \text{AP}_i \) is the average precision for query \( i \).

![Fig.6: Recall and Precision values for rankings](image3)

![Fig.7: Recall and Precision values for rankings](image4)
The MAP measure provides a very succinct summary of the effectiveness of a ranking algorithm over many queries. Although this is often useful, sometimes too much information is lost in this process. Recall-precision graphs, and the tables of recall-precision values they are based on, give more detail on the effectiveness of the ranking algorithm at different recall levels. Fig.6 and Fig.7 show the recall, precision graph for the two queries in the example. Graphs for individual queries have very different shapes and are difficult to compare.

Conclusion and Future Work:

Measured in terms of popularity, web search is clearly the most important search application. Millions of people use web search engines every day to carry out an enormous variety of tasks, from shopping to research. This work incorporates number of features in their ranking algorithms from the huge collection of user interaction data in the query logs. In this system the keywords and a query both are considered as vectors to compute similarity between them. This system used various similarities such as description similarity, functionality similarity, and characteristic similarity to find the relevant documents. The calculated similarity values are used for ranking the documents. Topical and functionality relevance has focused primarily to search required information in this system. Within a time user gets the information with the top most ranking data and the remaining top information will be moved to the data repository for future use. This design is to give good results for a range of queries, and better results for more specific queries. So the user in any environment can get correct information based on their desires. This work simulates that information retrieval on intelligent system, but actually the data recommendation is used to give justification for the intelligence. In future the same work can be extended purely on expert system without the intervention of the external user.

REFERENCES