IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 16, Issue 5, Ver. I (Sep – Oct. 2014), PP 67-78
www.iosrjournals.org

An Intelligent Approach to Information Retrieval System Using Enhanced DIG and FP-Tree Techniques

P. Janarthanan[1], N. Rajkumar [2]
[1]*(Department of Computer Applications, Sri Venkateswara College of Engg., Sri perumbudur, India)*
[2]*(Department of M.E., Software Engineering, Sri Ramakrishna Engineering College,Coimbatore, India)*

***Abstract**: Information retrieval is the process of retrieving all the relevant documents that satisfies the user query from large corpora. It is aimed to provide the relevant information and documents that matches the user query. Outcome of the several research results confirms that difficulties in information retrieval are matching the query with corpus. Consequently, the enhanced indexing technique named Document Index Graph (DIG) used for indexing document collection in order to match and retrieve information efficiently. Hence, an enhanced DIG has been constructed that stores all the stemmed sentences of documents in the graph. The words with same stem can be stored only once in DIG. This helps to reduce the size of the graph. The most frequently appearing words are planted into FP (Frequent Pattern) Tree. The FP-tree is a compact representation of all relevant frequently occurring information in a corpus. The enhanced FP tree with a table generates all types of possible term set which satisfy the minimum support. Information is retrieved with the help of FP-Tree and Document Index Graph.*

***Keyword**: Stemming, Document Index Graph, Query Processing, Frequent Pattern Tree and Information Retrieval.*

I. Introduction

Information retrieval can be described as representing the content of a document, representing User's information, ranking and relevance feedback. The document retrieval is one of the fastest growing and complex research areas in the field of information retrieval. An effective information retrieval can be obtained only under strong document retrieval methods. The user's information requirement is represented by a query or profile, contains one or more search terms and perhaps some additional information such as term weights. Hence, the information retrieval decision is done by comparing the terms of the query with the index terms appearing in the document itself. To have access to all the relevant documents related to a user query one need to have an information system which can efficiently and effectively retrieve them. Information retrieval systems are widely used to help user to find the required documents as per their needs [1]. In the field of information retrieval the following classic problem setting is studied: A user tries to satisfy an information need in a given collection of documents. The goal of the system is to retrieve documents with information content that is relevant to the user's information need [4]. Stemming plays an important role in indexing information prior to retrieving [2].

The document Retrieval is the computerized process of trading a list of documents that are appropriate to an inquirer's request by associating the user request to spontaneously produce an index of the textual content of documents in the system. There are a number of methods used for the retrieval of the documents such as clustering, indexing, page ranking, etc. In the field of information retrieval system, the relationship between a query and a document is determined primarily by the number and frequency of terms which they have in common [16].

An information retrieval system incorporating an indexing technique is used. The system is adaptive because it can retrieve the most relevant documents that satisfy the user queries. The document indexing is the key part of the concept, because indexing the documents helps in obtaining an interrelation between the documents. The document indexing also ensures the redundancy of documents in the database [4].

1.1 Data Mining

Data mining is an important aspect in Knowledge Discovery (KDD) in database process. It is an important concept in extracting unknown pattern or interesting knowledge from the very large volume of databases. Moreover, most of the data mining research problems involve retrieval of information from the database using various applications such as medical, image processing, text retrieval etc. Information retrieval is a subset of the data mining area in which sometimes user mines information by specifying the query with or without constraints [6].

1.2 Frequent Item set

The frequent sets play an essential role in many data mining tasks like mining association rules, correlations, classifiers and clusters. It tries to find interesting patterns from the corpus. The classification of item sets and products which often occur together in the given collection can be viewed as one of the most fundamental tasks in Data Mining. Let I = {I_1, I_2, .. , I_n) be a set of items. Let D, be the database that contains transactions. Each transaction T is a set of items and it is associated with an identifier TID. The occurrence frequency of an item set A is the number of transactions that contain the item set. If the occurrence of the item sets in the database is more than the minimum support count, then item set A is said to be a frequent item set [8].

1.3 Data Pre-Processing

A pre-processing step on input documents from document collection is to determine the representative index terms. It is the process of controlling the size of index terms and to improve the retrieval performance. It consists of special character removal, case conversion and stop words removal [1][12].

The first step is the special characters removal. It performs removing of special characters from the input text document. The lists of special character are represented in Figure 1.

```
!, @, #, ., ", \, $, %, ^, &, *, (, ), -, +, =,
_, {, }, [, ], ;, :, |, <, >, ?, /, ~, `, , , \
```

Fig.1. List of Special Characters

The second step is the case conversion. It converts all the upper case letters to the lower case letters for the input text document. The third step is the stop words removal. Stop words are the words that most frequently occurring but do not contain any meaning. It removes stop words from the text document. An eliminating stop words can improve processing and saves huge amounts of space in indexing. The Figure 2 shows the list of few stop words:

a, as ,to, but, when, give, very, that, among, just,
nothing, when, etc.,

Fig. 2. List of Stop Words

II. Related Work

Julie Beth Lovins and Anjali Ganesh analysed various stemming algorithms to reduce all words with the same stem to a common form. They explored several types of stemming algorithm with its merits and demerits. The classification of the stemming algorithms explained in detail. Comparisons between the stemming algorithms were performed. Julie proposed a new version of context-sensitive, longest match stemming algorithm called Lovins algorithm. The stemming and recoding procedures were explained [10].

A N K Zaman and Charles Grant Brown investigated on the performance of text retrieval systems. Latent Semantic Indexing Analysis (LSI/LSA) based ad hoc document retrieval task investigates the performance of retrieval systems that search a static set of documents using new questions. The Performance of LSI was tested using several small data sets. Three different term weighting schemes and own list of stop words have used to judge the performance. Recall, Precision and Coefficient of Variation were used to evaluate the retrieval performance of LSI based retrieval system [17]. Khaled M. Hammouda explored the concept of document indexing technique with more informative features including phrases and their weights that are important for indexing. A novel phrase-based document index model, the Document Index Graph (DIG) which allows incremental construction of a phrase-based index of the document set with an emphasis on efficiency rather than relying on single-term indexes only [12]. Walid Keirouz et al. proposed an information retrieval extensible approach which used natural languages to extract stem and minimizes the task of developing specific information retrieval system [16].

A.B.M.Rezbaul Islam and Tae-Sun Chung proposed a new algorithm and improved Frequent Pattern (FP) Tree with a table and a new algorithm for mining the association rule. It described about mining the entire possible frequent item set without generating the conditional FP-Tree. It also provided several ways of finding the frequency of the frequent items which was used to estimate the desired association rule [15]. Jiawei Han et al. proposed an efficient method of mining frequent item set without candidate generation. It minimized time and saves memory space. They developed an efficient FP-Tree based mining method. It also provides a performance study that shows that the FP - growth method is efficient and scalable for mining both long and short frequent items and is about an order of degree faster than the apriori algorithm and also faster than some recently reported new frequent pattern mining methods [7][9].

The uninterrupted texts hunt Queries implemented by Kyriakos Mouratidis and HweeHwa Pang. In this research, solution for processing continuous text queries proposed efficiently. The objective was to support a large number of user queries while sustaining high document arrival rates. This solution indexes the streamed documents in main memory with a structure based on the principles of the inverted index file and processes document appearance and expiration events with an incremental threshold-based method [13]. Fatiha Boubekeur et al. proposed the association rule constructs and ontology concepts for semantic document indexing. They described a novel approach for document indexing based on the discovery of contextual semantic relations between concepts. Finally, concepts and related contextual relations were organized into a conditional graph [6].

Ke-Chung Lin, I-En Liao and Zhi-Sheng Chen proposed the improved frequent pattern (IFP) growth algorithm. In their research, the IFP-growth (Improved FP-growth) algorithm used to improve the performance of FP-growth. IFP-growth employs an address-table structure to lower the complexity of mapping frequent 1-item sets in an FP-tree. It used a hybrid FP-tree mining method to reduce the need for rebuilding conditional FP-trees [11]. B. Barla Cambazoglu and Cevdet Aykanat implemented efficient query processing techniques to find the distinct terms over a large document collection and they used inverted indices to index documents [3]. Daniel Osuna-Ontiveros et al. proposed a topic based indexing approach to represent topics associated with documents. The documents were modeled by using clustering algorithms based on natural language processing. As a consequence of this proposal, the document-topic matrix representation denoting the importance of topics in the document was implemented. In this work, each query over documents was converted into a vector of topics. Similarity measure can be applied over this vector and the matrix of documents to retrieve the most relevant documents [4]. Djamal et al. introduced improved indexing technique for whole document retrieval [5]. Liang Wang and David Wai-Lok Cheung represented how to mine the frequent item in large scale and uncertain dataset. It is a task of handling huge items for all data mining phases [14].

III. Proposed Work

In our proposed work, three main concepts are used here namely: Indexing, Frequent Pattern Tree generation and Information Retrieval. Prior to these concepts, there are some pre-processing steps which include the most common document pre-processing techniques like special characters removal, stop words removal and the stemming. The input text documents are pre-processed to remove the special characters available in the document. The stop words that are commonly utilized words that carry less important meaning in the text document are also removed from the text document. The case conversion is performed for converting the upper case letters into lower case letter. Word stemming is an important feature that has been developed for information retrieval in order to reduce morphological variants to their root form. Most of the words in a text document have various morphological variants. Since the variants have a similar semantics they can be considered as equivalent for the purpose of many retrieval tasks. Stemming is the process of reducing a word to its stem, and a stemmer or a stemming algorithm is a computer program that automates the task of stemming.

The document representation is usually called as indexing. The main objective of indexing is to assign to each document a descriptor represented with a set of features, usually keywords, derived from the document content. The Document Index graph (DIG) is used for indexing the documents. DIG indexes the documents while maintaining the sentence structure. An enhancement has been performed with DIG such that the stemmed words are stored in the graph. As a result, words with same stem are stored only once in the graph. This reduces the size of the document index graph. DIG records the term frequency (TF) of the word in that document. Nodes in the graph carry information about the documents they appeared in, along with the sentence path information. The sentence structure is maintained by recording edges between the words. The appearance of the particular word along with the document number, sentence number and word number are maintained in the index graph. The FP-growth method proposed a data structure called the FP Tree. Every branch of FP-Tree represents a frequent item set and the nodes along the branches are stored in decreasing order of frequency of the corresponding items with leaves representing the least frequent items. An enhanced FP-Tree consists of mainly two elements: the tree and a table. The tree shows the relation among the items more specifically and table is used to store the spare items. It is stored in spare table which has two columns namely item name and frequency. In enhanced FP-Tree, the stemmed words are stored in nodes of the tree. This overcomes the drawback of traditional FP-Tree in which a group of branches is created and the same item occurs several nodes. All the infrequent items are stored in the hash table.

Once the user query is entered, query processing is completed to remove the special characters and stop words in the given query. After stop word removal, the words in the query are stemmed. The stemmed words are then checked with FP-Tree to see if it is a frequently occurring item. If it is found in FP-Tree, the relevant documents and sentences are retrieved using the DIG. If not found in FP-Tree, the word can be searched with hash table and its relevant documents and sentences are retrieved from Document Index Graph. The Figure 3 shows the proposed system.

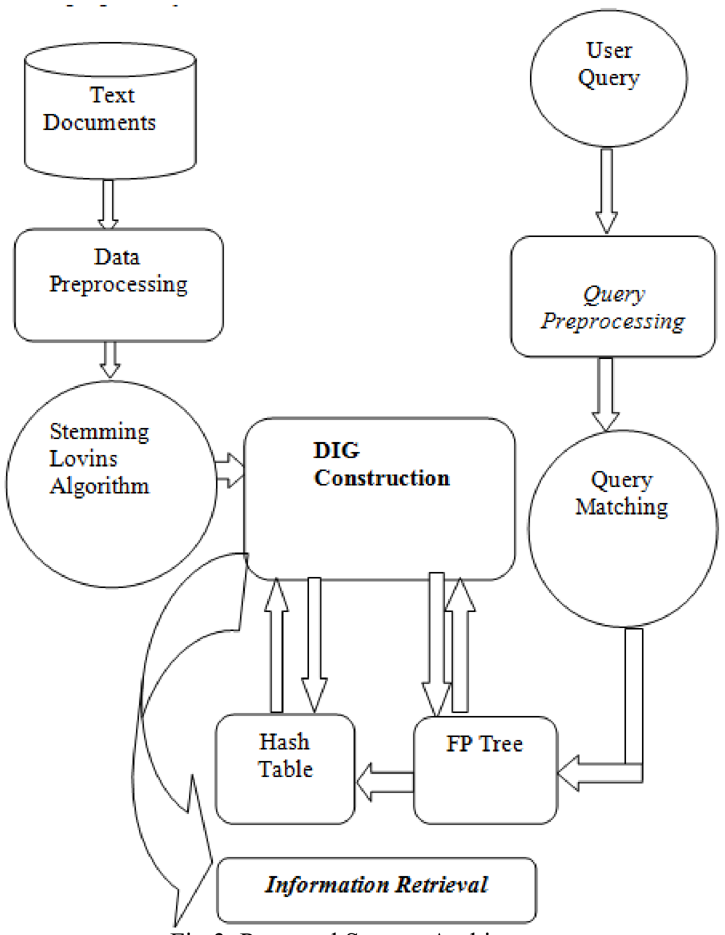


Fig.3. Proposed System Architecture

3.1 Lovin's Stemming Algorithm

Lovins stemming algorithm is a single pass stemming algorithm. Lovins is a context sensitive stemmer. This algorithm removes endings based on the longest-match principle. The Lovins stemmer removes a maximum of one suffix from a word due to its nature as a single pass algorithm. It applies a catalogue of about 250 different suffixes and removes the longest suffix attached to the word ensuring that the stem after the suffix has been removed is always at least 3 characters long. Then, the stem ending may be reformed by referring to a list of recoding transformations.

The stemming algorithm has 294 endings, 29 conditions and 35 conversion rules. Each ending is associated with one of the conditions. Lovins algorithm consists of two major processing steps:

- If the lengthiest suffix is found which satisfies its associated condition and is removed.
- In the second step, the 35 rules are applied to transform the ending.

The Table 1 lists the few endings which are used in processing step 1. They are grouped by length from 11 characters down to 1 character. Each ending is followed by its condition code.

Table I. List of Endings

.11.			
alistically B	arizability A	izationally B	
.10.			
antialness A	arisations A	arizations A	entialness A
.09.			
allically C	antaneous A	antiality A	arisation A
entations A	entiality A	entialize A	entiation A
izational A			
.08.			
ionality A	ionalize A	iousness A	izations A
.07.			
ability A	aically A	alistic B	alities A
atingly A	ational B	atively A	ativism A
entiate A	entness A	fulness A	ibility A
ication G	icianry A	ination A	ingness A
iteness A	iveness A	ivistic A	ivities A
.	.	.	.
.01.			
a A	e A	i A	o A

Table 2. Codes for Context Sensitive Rules

Condition Name	**Rule Description**
A	No restriction on stem
B	Minimum stem length = 3
C	Minimum stem length = 4
D	Minimum stem length = 5
E	Do not remove after ending e
F	Minimum stem length = 3 and do not remove after ending 'e'
....	....
V	Remove ending only after c
W	Do not remove ending after s or u
X	Remove ending only after l, i or u*e
Y	Remove ending only after in
Z	Do not remove ending after f
AA	Remove ending only after f, l, er, d, or, es or t, th, ph
BB	Minimum stem length = 3 and do not remove ending after met or ryst
CC	Remove ending only after l

The following are the 29 conditions called A to Z, AA, BB, CC and * stands for any letter. These are codes for context-sensitive rules associated with certain endings as in Table 2. There are 35 transformation rules used in recoding stem terminations which are represented in Table 3. This step is done compulsorily without considering whether or not an ending is removed in the first step.

Table 3. List of Transformation Rules

R.No.	**Transformation Rule**	**R.No.**	**Transformation Rule**
1	remove one of double b,d,g,l,m,n,p,r,s,t	19	uad → uas
2	iev → ief	20	vad → vas
3	uct → uc	21	cid → cis
4	umpt → um	22	lid → lis
5	rpt → rb	23	erid → eris
6	urs → ur	24	pand → pans
7	istr → ister	25	end → ens except following s
13	pex → pic	31	ent → ens except following m
....	....		
14	tex → tic	32	ert → ers
15	ax → ac	33	et → es except following n
16	ex → ec	34	yt → ys
17	ix → ic	35	yz → ys
18	lux → luc		

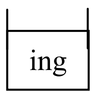
Let us assume word raining to be stemmed.

Step 1: Letters from the last position are taken. Scan the list of endings in Table 1 and its rules in Table 2, until minimum stem length is reached. The example of stemming is given in Example 1.

Example 1. Stemming Example

Letters	Endings	Availability
g	01	Not Found
ng	02	Not Found
ing	03	Found (N)
ning	04	Not Found

Put in stack



Minimum stem length is now 3. According to rule N,
raining $\rightarrow$ rain
Step 2: Scan the transformation rules. No rule found.
Therefore, final stemmed word from word raining is rain.

3.2 Document Index Graph

The proposed Document Index Graph (DIG) indexes the documents while maintaining the sentence structure. The DIG is a directed graph (digraph) G= (V, E) Where, V is a group of nodes $\{v_1,v_2,\ldots,v_n\}$ where each node v represents a unique word in the entire document set and E is a set of edges$\{e_1,e_2,\ldots,e_m\}$such that each edge e is an ordered pair of nodes(v_i,v_j). An edge (v_i, v_j) is from v_i to v_j and v_j is adjacent to v_i. There will be an edge from v_i to v_j if and only if the word v_j appears subsequent to the word v_i in any document. The above definition of the graph suggests that the number of nodes in the graph is same as the number of unique words in the document collection. Assume that a sentence of m words appearing in one document consists of the following word sequence $\{v_1, v_2,\ldots, v_m\}$. The sentence is represented in the graph by a path from v_1 to v_m such that $(v_1, v_2), (v_2, v_3),\ldots, (v_{m-1},v_m)$ are edges in the graph. The Document Index of DIG is mathematically represented as in Equation (1).

Document Index

$$(DI) = \{t_{id}, t_n, tf, s_{id}, s_n, d_{id}, sen_n, w_n, e_{id}, p\} \quad (1)$$

Where $\mathbf{t_{id}}$ is the unique id of the term in the document, $\mathbf{t_n}$ is the name of the term, **tf** is the frequency of each term occurring in all the documents, $\mathbf{s_{id}}$ is the unique stem id of the respective term $t_{id,}$ $\mathbf{s_n}$ is the name of the respective stem id $s_{id,}$ $\mathbf{d_{id}}$ is the document number in which the term t_{id} occurs, $\mathbf{sen_n}$ is the sentence number in which the term t_{id} occurs in the document, $\mathbf{w_n}$ is the word number in which the s_{id} resides in the document, $\mathbf{e_{id}}$ is the unique edge id which maintains the sentence structure and **p** is the previous term .

Now, let us consider the term "raining" whose stem is "rain" in the 1st sentence, 4th word of the 1st document whose previous term is "season" and has appeared 4 times in the documents.

$$DI = \{ 21,raining,4,10,rain,1,1,4,3,season\}$$

Our proposed document index graph contains only one node for words that share a common stem word. The illustration of the proposed structure is represented in Fig. 4 includes term table which maintains multiple words of a stem word. It overcomes the problem of having several nodes for words of the same meaning. The documents before data pre-processing and after pre-processing is used here to demonstrate the proposed node and term table in DIG.

The proposed DIG structure maintains three types of tables namely - Term table, Document table and Edge table. Each term entry in the term table records the term name and its term id. Each document entry in the document table records the term frequency of the word in that document. Since the graph is directed, each node maintains a list of an outgoing edge per document entry. This set of edges indicates that which phrase continues along the edge. Each entry in an edge table records the edge number, sentence number and word number thus maintaining the sentence structure of the document. The edge table is a list of an outgoing set of edges E_d.

The size of the document index graph can be found using the following formula:

$$Size (G) = 3 (m * df_{avg}) + q * n \quad (2)$$

Where n is the number of documents in the data set, m is the number of unique terms in the data set, df_{avg} is the average inverse document frequency and q is the average number of terms per document. The Figure

4 is the model of the proposed document index graph and Figure 5 shows the DIG for sample documents. Each sentence in a document is taken as a single transaction in the database. The sentences from different documents are distinguished using different line styles.

rain drop water earth surfac rain major compon water cycl **Document 1**	rain season veges grow substant surfac run harvest roof top harvest **Document 2**
rain season agricultur harvest rain water harvest **Document 3**	

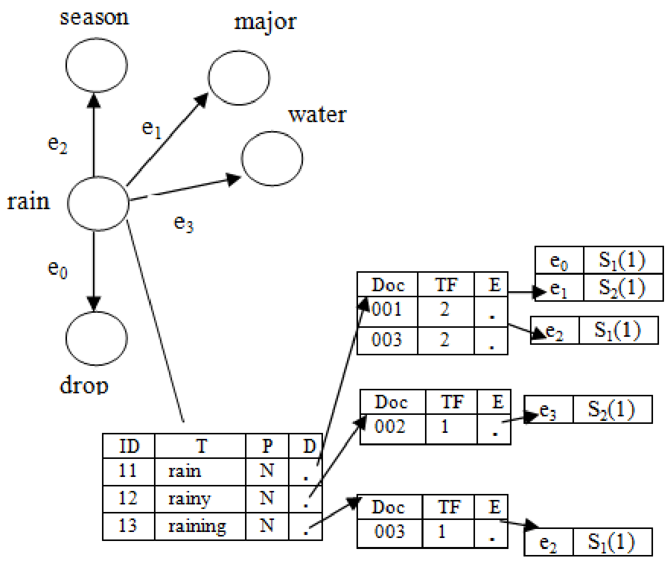


Fig. 4. Proposed DIG Structure for node Rain

rain harvest surfac rain harvest rain harvest earth	major rain Drop rain harvest water cycl	agricultur earth agricultur water cycl veges agricultur earth rain agricultur
Document 1	**Document 2**	**Document 3**

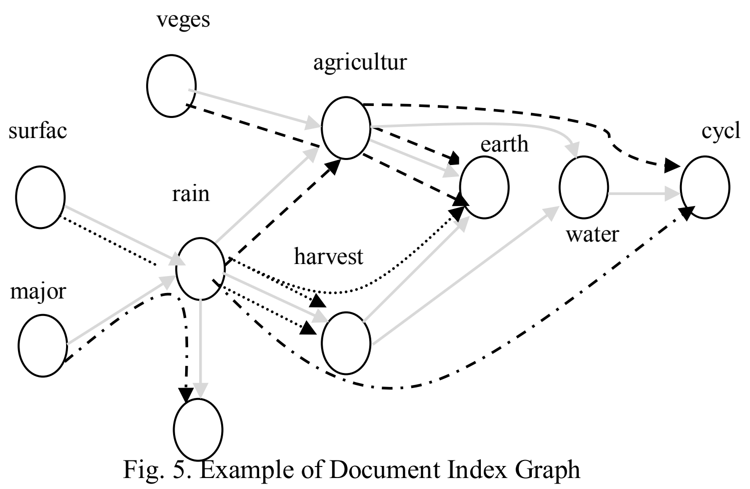


Fig. 5. Example of Document Index Graph

3.3 Enhanced Frequent Pattern Tree

In order to speed up the retrieval process, the FP-tree approach has been used to maintain the frequently occurring terms in the documents. Our proposed FP-Tree consists of mainly two elements such as the table and a tree. The tree depicts the correspondence among the items more specifically and table is used to store the infrequent items. It is called as spare table which has two columns namely item_name and frequency. An item name is the name of the items and frequency means how many times it occurs in spare table. The main notion to introduce the spare table is to minimize the branches created for items that occur multiple times. The spare table can store more than one time occurring items. But in our proposed FP-tree, every distinct item has one node. Hence, it is easier and efficient for further processing. There is a threshold on support count to improve efficiency of the retrieval system. As in enhancement, the top two highest support counts are considered as most frequent items to fetch information based on it. This leads to the balanced size of FP-tree and hash table.
The walkthrough example of finding frequent terms using FP-Tree approach:

Table 4. Pre-processed Document

ID	Transactions
1	rain drop water earth surfac cloud
2	rain major compon water cycl
3	rain season veges grow substant
4	surfac run harvest
5	roof top harvest
	rain season agricultur harvest
7	rain water harvest

Table 5. Term Occurrences in the document

ID	Term	Frequency
1	Rain	5
2	Drop	1
3	Water	3
4	Earth	1
5	Surface	2
6	Cloud	1
7	Major	1
8	Compon	1
9	Cycl	1
10	Season	2
11	Veges	1
12	Grow	1
13	Substant	1
14	Run	1
15	Harvest	4
16	Roof	1
17	Top	1
16	Agricultur	1

Table 6. Items – Satisfy Minimum Support count

ID	Term	Frequency
1	Rain	5
3	Water	3
5	Surface	2
10	Season	2
15	Harvest	4

First, scan the transactions and calculate the number of occurrences of each individual term as in Table 5. Then compare frequency of each term with the minimum support count. Assume that minimum support count is 2. The terms that satisfy the minimum support count can be considered to construct the frequent pattern tree as in Table 6. These terms are said to be frequently occurring term. The terms that do not satisfy the minimum support count are not considered to construct the frequent pattern tree in Table 7. They are stored in the hash table. These terms are said to be infrequently occurring terms. The transactions with frequently occurring terms are alone taken in the next step as in Table 8. The transactions are sorted in descending order using the count that is in Table 9. The descending ordered transactions are two Types:

Type 1: Transactions with frequent term.
Type 2: Transactions without frequent terms.

Table 7.Terms – Not Satisfy Min. Support Count

ID	Term	Frequency
2	Drop	1
4	Earth	1
6	Cloud	1
7	Major	1
8	Compon	1
9	cycl	1
11	veges	1
12	grow	1

Table 8. Transaction Items in Original Order

ID	Transactions
1	rain water surface
2	rain water
3	rain season
4	harvest surface
5	Harvest
6	rain harvest season
7	rain harvest water

Table 9.Transaction Items in Descending Order

ID	Transactions
1	rain water surface
2	rain water
3	rain season
4	surfac harvest
5	Harvest
6	rain season harvest
7	rain water harvest

13	substant	1
14	run	1
16	roof	1
17	top	1
16	agricultur	1

Here, all the transactions start with the top two most frequently occurring items. Hence all the transactions start with the frequent pattern tree as in Fig. 6. The top most frequent item is "rain" and second most frequent item is "harvest". If a transaction does not start with most frequently occurring items, then it is added into the FP-Table X.

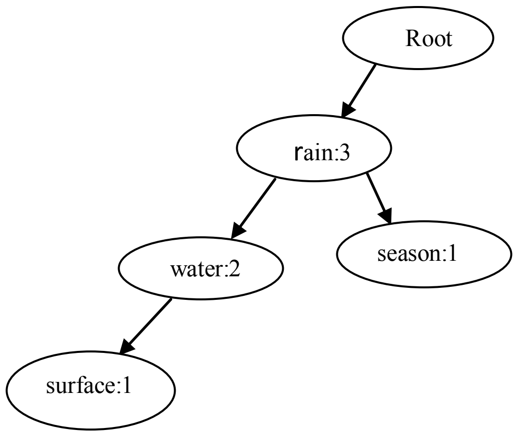


Fig. 6. FP-Tree Structure

Table 10. FP-TABLE

Item	Count
Surface	1
Harvest	2
Water	1
Season	1

The transactions that do not start with the most frequent item are stored in FP-Table. The items that are repeated while inserting into FP-Tree are also stored in FP-Table so as to avoid redundancy. The infrequent items that do not satisfy the minimum support count are put into the hash table with starting alphabet as the key and the term as the value. The steps to be followed to store the infrequent items in the hash table for the above explained example are as follows:

For Terms:	**For Digits:**
ASC >= 97 and ASC <= 122 [letter]	ASC >= 98 and ASC <= 57 [digit]
TreeSet Index = ASC-97	TreeSet Index = ASC-48+26
Hash Key= Character (TreeSet Index + 97)	Hash Key= Character (TreeSet Index+48-26)

3.4 Efficient Information Retrieval

The input documents are pre-processed to remove stop words and special characters. The pre-processed documents are then stemmed which is stored in the document index graph. The document index graph maintains the document number, sentence number and word number of each term. The most frequent items are stored in the FP-Tree and the infrequent items are stored in the hash table. The input query is first pre-processed. The stemming task is performed after query pre-processing. It is compared with the terms in the frequent pattern tree. If the relevant documents are found, then the respective documents can be retrieved from the DIG.If the term does not match with FP-Tree then the hash table is searched .If a match is found, then the related documents are retrieved from DIG. Here, not only the documents retrieved but also the sentences that contain the given query also retrieved using the DIG as in Fig. 7.

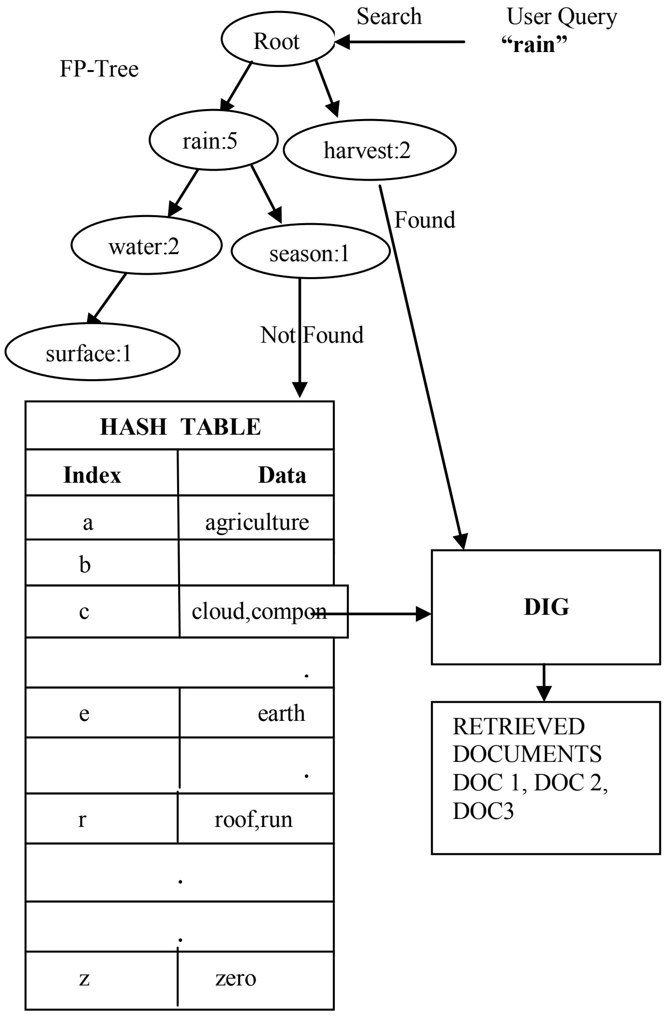


Fig. 7. Proposed Information Retrieval System

The proposed system has several advantages. First, stemming has been performed using lovins algorithm. It is a single pass and has only two major steps that stem all double letters also. So, it reduces time to stem the words. The document index graph is constructed using stemmed words. The enhanced Document Index graph maintains the sentence structure by recoding three tables namely term table, document table and edge table. The term table is used to store variant terms of stem word. So, it eliminates number of nodes to be created. The DIG based retrieval system, not only retrieves the documents but also the sentences related to the user query.

IV. Experimental Results

In order to evaluate the performance of information retrieval using document index graph and frequent pattern tree, the model of the system was developed to study the performance system. The system was developed in Java Net Beans and MS-Access used for the database. The elaborate experiment was conducted to measure document indexing and graph construction on our system using benchmark documents collected from Wikipedia. Microsoft Access database was used as back-end to store the document index table objects. All the modules have been developed one by one separately to calculate the efficiency of the system. The FP-Tree construction time increases as more number of unique words appears in the document collection as in Figure 8.

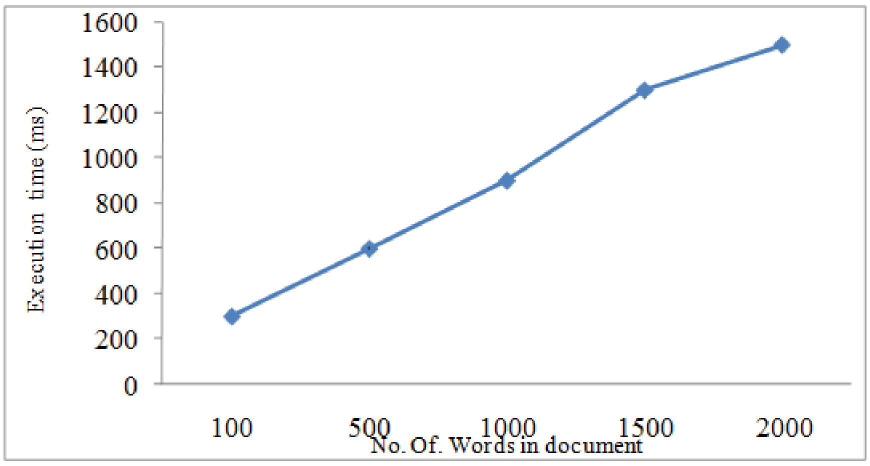


Fig. 8. Time Taken To Construct FP-Tree

During the construction of DIG, it takes time to construct index graph when the corpus having more and multiple words of a stem word as in Figure 9.When the same word occurs more than one time, the frequency of the word can be incremented and sentence number is updated in edge table. The information retrieval system performance is represented in Figure10. It retrieves more information when a user provides several query terms.

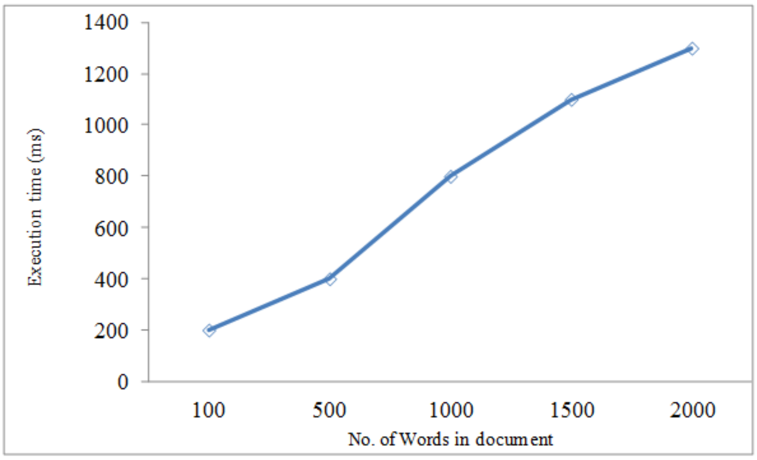


Fig.9. DIG Construction Time

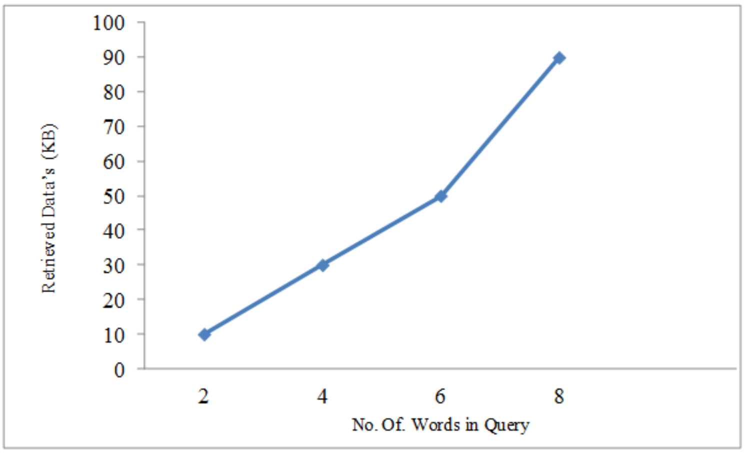


Fig. 10. Amount of Information Retrieved

The following graph in Figure 11 is to retrieval time using DIG verses retrieval using DIG and FP-Tree.This graph compares the time taken to retrieve the documents using DIG and DIG with FP-Tree.

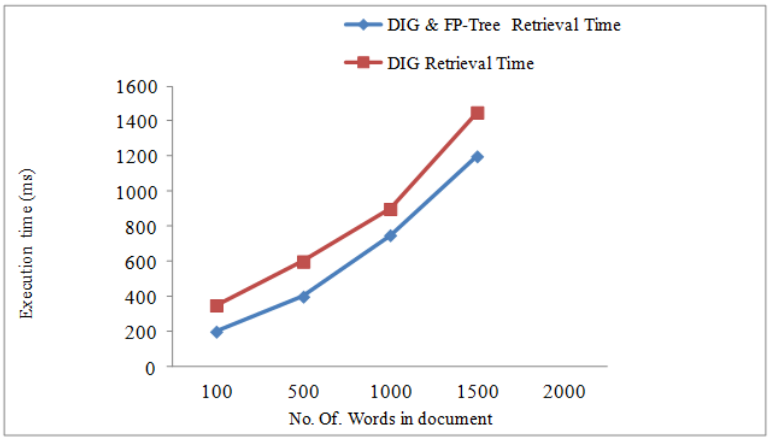


Fig. 11. Comparison of Retrieval Time

V. Conclusion

The document index graph (DIG) is the efficient way of converting documents into graph which consist of root words in the documents. It facilitates efficient and fast retrieval of information and relevant documents. The enhanced document index graph structure stores the stemmed terms which reduces the size of the graph when several terms share the same stem word. Combining FP-Tree with document index graph improves the information retrieval process and provides the exact information quickly because frequently occurring terms in the documents were constructed as a FP-Tree. The information retrieval process using Document index graph provides complete information to user and also it fulfils phrase completion.

The FP-Tree method reduces the search time and speed up the retrieval process by providing frequent terms in the corpus. So, the response time of system to user query is very quick. Our proposed method is well-organized technique of improving document index graph and FP-Tree in order to provide an efficient information retrieval system. Future, ontology based concept can be used to improve the information retrieval process further.

References

[1] Abebe Rorissa and Xiaojun Yuan, “Visualizing and Mapping the Intellectual Structure of Information Retrieval”, Information Processing and Management, Vol.48, No.1, 2012,120–135.
[2] Agbele K. et al “Context-Aware Stemming Algorithm for Semantically Related Root Words”, African Journal of Computing & ICT, Vol. 4,No.5, 2012, 33-42.
[3] Barla Cambazoglu B and Cevdet Aykanat, “Performance of query processing implementations in ranking-based text retrieval systems using inverted indices”, Information Processing and Management, Vol.42, No.4,2006, 875–898.
[4] Daniel et al, “A topic based indexing approach for searching in documents”, 8th International Conference, Electrical, Merida, 2011, 1-6.
[5] Djamal Belazzougui, Gonzalo Navarrob and Daniel Valenzuela, “Improved compressed indexes for full-text document retrieval”, Journal of Discrete Algorithms, Vol.8, 2012, 3-13.
[6] Fatiha Boubekeur, Mohand Boughanem and Lynda Tamine - Lechani, “Exploiting association rules and ontology for semantic document indexing “, Proceedings of 12th International Conference on Information Processing and Management in Uncertainty-Based System, 464-472, Malaga, Spain, 2008.
[7] Jianyong Wang, Jiawei Han, Ying Lu, and Petre Tzvetkov, “An Efficient Algorithm for Mining Top-K Frequent Closed Item sets”,Knowledge and Data Engineering, Vol. 17, No. 5, 2005, 652-664,.
[8] Jiawei Han and Micheline Kamber, Data mining Concepts and techniques, (3rd Edition, 2011).
[9] Jiawei Han, Jian Pei and Yiwen Yin, “Mining Frequent Patterns without Candidate Generation”, Data Mining and Knowledge Discovery, Vol. 8, No.1, 2004, 53–87.
[10] Julie Beth Lovins, “Development of a Stemming Algorithm”, Mechanical Translation and Computational Linguistics, Vol.11, No.1 and 2, 1968, 22-31.
[11] Ke -Chung Lin, I-En Liao and Zhi-Sheng Chen,” An improved frequent pattern growth method for mining association rules”, Expert Systems with Applications Vol.38, No.5, 2011, 5154-5161.
[12] Khaled Hammouda M and Mohamed Kamel S “Efficient Phrase-Based Document Indexing for Web Document Clustering “,Knowledge and Data Engineering, Vol. 16, No. 10, 2004, 1279-1296.
[13] Kyriakos Mouratidis and HweeHwa Pang, “Efficient Evaluation of Continuous Text Search Queries”, Knowledge and Data Engineering, Vol. 23, No. 10, 2011, 1469-1482, .
[14] Liang Wang, David Wai-Lok and Reynold Cheng, “Efficient Mining of Frequent Item Sets on Large Uncertain Databases”, Knowledge and Data Engineering, Vol. 24, No.12, 2012, 2170-2183.
[15] Rezbaul Islam M and Tae-Sun Chung,” An Improved Frequent Pattern Tree Base Association Rule Mining Technique”, Proc. of International Conference on Information Science and Applications, Jeju Island,Korea, 2011, 1-8.
[16] Walid Keirouz,Haider Harmanani and Saeed Raheel,“A Rule Based Extensible Stemmer for Information Retrieval with Application to Arabic”, The International Arab Journal of Information Technology, Vol.3,No.3,July, 2006.
[17] Zaman A and Charles Grant Brown**, “**Latent Semantic Indexing and Large Dataset: Study of Term-Weighting Schemes**”,** IEEE, 5th International conference Digital Information Management, Lakehead University, Canada, 2010, 1-4.