*International Journal of Computer Applications (0975 – 8887)*
*Volume 58– No.20, November 2012*

# Information Retrieval System Assigning Context to Documents by Relevance Feedback

Narina Thakur
Department of CSE
Bharati Vidyapeeth College Of Engineering New Delhi, India

Deepti Mehrotra
ASCS Amity University, Uttar Pradesh, India

Abhay Bansal
Department of CSE, ASET
Amity University
Uttar Pradesh, India

## ABSTRACT

In this paper we have proposed user feedback driven Information retrieval model. The proposed model assigns weights to the retrieved documents based on its context. The documents are re-ranked based on the user profile and his feedback. Proposed Information retrieval system uses vector space model and expert system. Need for user profile and relevance of information while searching and extracting information, from information retrieval system is highlighted.

## General Terms

Information, retrieval, relevance, feedback

## Keywords

Information Retrieval, Relevance Feedback, Vector Space Model, Inverted Index.

## 1. INTRODUCTION

Today with the emergence of digital library and electronic media exchange, Information overload is day by day becoming a vast concern in Information retrieval. Automated information retrieval systems can be one solution. Information retrieval (IR) has evolved in the past; various new strategies has evolved in the area of sourcing and gathering information. Information retrieval systems should ensure quality and relevant information resources retrieved from a collection of information resources. Researches in the area of metadata or on full-text indexing are emerging. Digital / public libraries and web search engine uses Information retrieval system to provide access to books, journals and other web documents. Information retrieval is finding information of an unstructured nature that satisfies an information need from within large collection.

IR lacks improved information retrieval mechanism to extract relevant information for user query. Existing Information retrieval systems are not user profile guided systems, hence the search engine result, huge irrelevant information, which leads to low precision rate. For e.g. when a user enters a search query Cloud burst. Search results in information about Cloud burst issues in Cloud computing, Cloud bursting technology, cloud security and the desired information about the Cloud burst a natural calamity is not retrieved. Hence user is getting inadequate information, even with existing recommender systems which requires human intervention in the search process. The given solution can be suitably used for Semantic LOR. With the dearth of relevant information extraction system, Information retrieval is day by day becoming a cumbersome process. Thousands of documents are returned every time for a particular query search.

Section II elucidates the Information retrieval system, issues and challenges faced by an Information Retrieval system. Section III introduces relevance based Information Retrieval system. Section IV and Section V propose Feedback reweights to keywords using expert system in IR architecture and framework for Information retrieval.

## 2. Information Retrieval System

Information retrieval system is concerned with the structure, analysis, organization, searching, retrieval and storage of information. Information retrieval system retrieves information that satisfy user requirement from corpus.
It can also be defined as retrieval of relevant documents based on user requests, commonly called as user queries.

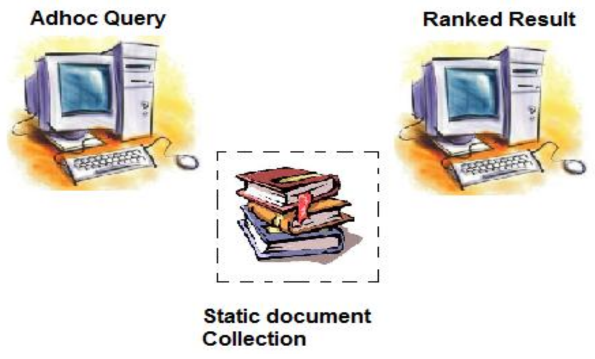


**Fig 1 illustrates the basic process of information**

Some common issues and challenges in Information retrieval from any corpus are:

- **User and their context:** user and information is one of the major challenges, as search evaluation is user centered. Keywords queries are poor description of actual information need. For effective IR system, user interaction and semantics plays a key role for extracting relevant information and it will help the IR to understand user intent.

- **Multiple language and media:** Cross language information retrieval is information retrieval is one of the major challenges. The users submit the query in one language and get retrieved documents in another language. One assumption is that user understands results in multiple languages or Information retrieval system should converts foreign language to user understandable language format.[1]

- **Clearer task definition:** Finding relevant documents and not just finding the simple matched document to pattern. For e.g. Polysemy associate one term with two or more dissimilar senses. “Dogs would always bark at strangers” or “The tree's bark was a rusty brown”.
- **Acquisition of better learning algorithm and more training data:** Learning algorithms must be enhanced using techniques based on relevant information and large feature training data.
- **Improved formal model:** Numerous information retrieval models exist but since users have become complacent in their expectation of accuracy in information retrieval, so improved formal information retrieval model are required.

## 3. Proposed Information Retrieval system Architecture

Document can be a web page, email, book, stories, test documents, IR sessions, Text messages, PPT, Word doc, PDF, patents, Postings, Forums etc. Significant text contents and document structures (e.g. Title, Author, date for document; subject, sender and destination for emails) are the salient properties of a document.

Relevance is a statistical property of text rather than linguistic. Relevance can be expressed in terms of task, context, novelty and style in a user based model. Relevance and ranking algorithms are recommended to augment information retrieval process. Keywords of queries are poor description of actually information need. [2]User is sometimes ignorant of the actual information need. The relevance of the document can be enhanced by query expansion, query suggestion and by incorporating expert system in relevance feedback for improved document ranking.

Here we have proposed a user feedback based personalized expert Information retrieval system. The system is based on Vector Space model. Vector Space model face the limitation that large documents will likely have higher probability to be ranked high than small one, due to frequency of keywords or terms. Here expert system is used which can be a user, ontology or Neural network. User interaction in retrieval process will enhance search results.

In this research proposal dynamic information retrieval system is proposed where initially documents will be ranked based on vector space model. Information retrieval system efficiency can be enhanced by using user feedback and expert system based search guidance.

## 4. Proposed Information Retrieval system Components

Information retrieval system architecture consists of mainly two components- Indexer and Query processor. The indexer creates an inverted index / Dictionary of keywords for documents. It will enhance the document processing speed for retrieving. User submit query to Query processor which takes user query and retrieves ranked list of relevant documents.

### 4.1 Indexer

#### *4.1.1 Preprocessing*

Documents are initially collected and stored in corpus. The document as a whole is very large to be operated upon, text is first tokenized, [3] content dictionary is prepared [4], followed by preprocessing of the text like stop-word removal and Stemming [5]. Stemming reduce the number of keywords by removing the common words like ‘look’, looking’ and ‘looked’ into one word ‘look’.

#### *4.1.2 Index Construction*

All the information retrieval system uses inverted index instead of sequential scan of the whole document corpus. Inverted index helps in mapping keywords to documents list. Inverted index can become convenient by dropping the number of keywords.

All the documents in the document corpus are assigned a unique doc id. Fig2. Shown below describes the basic process of creating inverted index. [6]

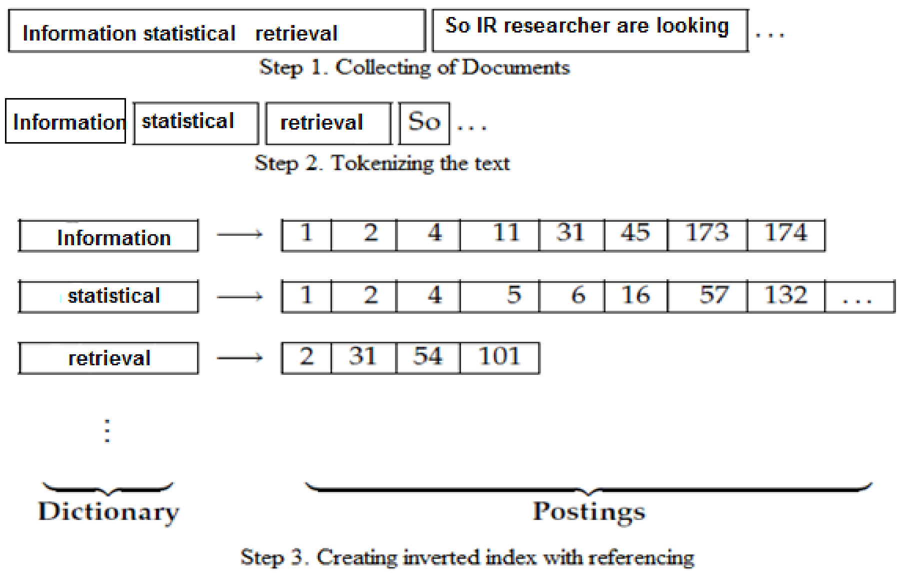

Information statistical retrieval
So IR researcher are looking
. . .
Step 1. Collecting of Documents
Information
statistical
retrieval
So
. . .
Step 2. Tokenizing the text
Information
1 2 4 11 31 45 173 174
statistical
1 2 4 5 6 16 57 132 . . .
retrieval
2 31 54 101
Dictionary
Postings
Step 3. Creating inverted index with referencing

**Fig2. Basic process of creating inverted index**

### *4.1.3 Index Compression*

To reduce the number of I/O operations required to build, query the inverted index and for storage overheads, inverted indices are compressed using various compression algorithms. [7]

### ***4.1.4** Weight Assigning to keywords & Vector Space **Model***

After creating the inverted index, we calculate the weights of each tokens by using Inverse document Frequency (IDF) metric, so that the vector can be calculated for further processing in query processor through vector space model. [8, 9]

## 4.2 Proposed Query Processor

### *4.2.1 Query Processing*

For query processing, the user will login into the system. Each user account will have a proficiency level metric associated with it. Proficiency level metric will be used for the relevance feedback in which documents will be ranked based on Similarity Coefficient (SC) and proficiency level of the feedback user and expert system.

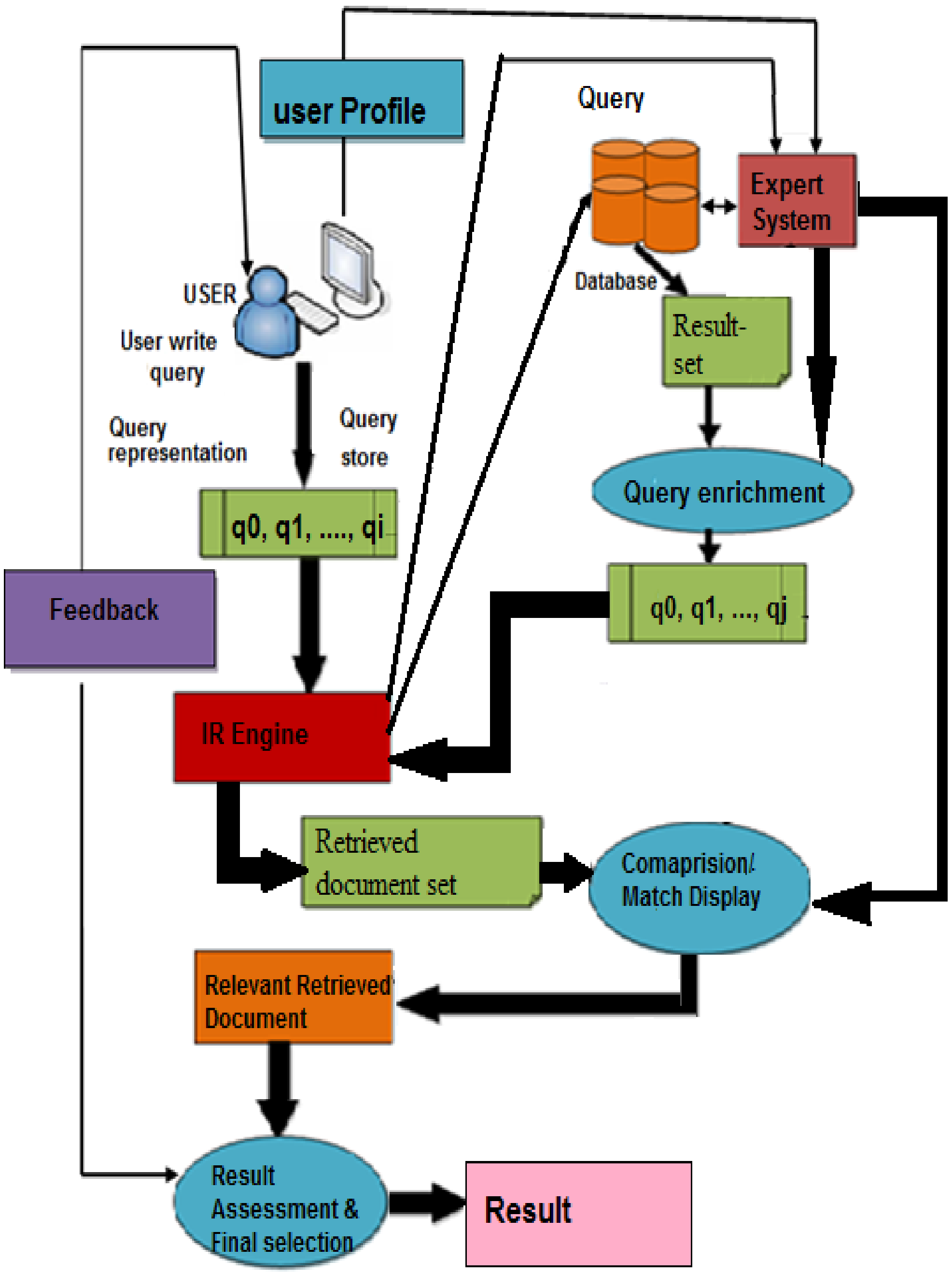


**Fig.3. Feedback reweight to keywords using expert system.**

### *4.2.2 Feedback Reweight to Keywords*

User feedback will be taken into consideration based on the clicked or explored documents (known as positive feedback) for retrieved documents. User relevance can also be improved by incorporating Expert system in the retrieval process. User profile is given as an input to the Expert system. Expert system can be Ontology or it can be a neural network built on past experience of varied user applied for searching documents. Expert system will function in two modes: retrieval of relevant result set using past history of document search and profile based relevance.

Basically, we are considering the expertise of the user to rank the documents, to judge the level of the document for both user feedback and frequency of keywords in the document. The document corpus domain will be based on user profile selection. Given the document that the user has visited, IR system recommends other documents with contents that are similar to the previously viewed documents. Here we propose to use user ontology profile for extracting the interests from user profile and subset of the user's activities records from visited documents.

### *4.2.3 Query Modification*

The user context is considered to judge the relevance of user query. Semantic similarity was considered as a function which took arguments as user query and document corpus (c) and calculate semantic relevance of document (d).

Semantic Relevance(c, q) =dR $\in$ R where q is query and R is a set of real numbers.

The co-domain semantic relevance(c, q) is therefore made up of real number R (in interval of 0...1) interpreted as semantic relevance (c1, q) < semantic relevance (c2, q) represent the fact that c1 is less relevant than c2 with respect to user query q and the user context.

To calculate the semantic similarity among query and document, document and an element from set Ci and Q. The similarity between keyword in the query and each keyword in the document was then calculated and the sum of the closest match gives the overall similarity. e.g.

Given two sets Q and Ci where Q= (qw1, qw2......., qwn) and

Ci= (ciw1, ciw2........., ciwm)

where m and n are number of keywords in the user entered query and number of distinct corpus respectively. The similarity between Q & Ci is calculated using Rocchio algorithm. [10].

## 5. Framework for proposed Information retrieval system

Text mining involves computations on text to gain interesting information. Keyword document matrix is created which contain frequency of distinct keywords of each document; other approaches can be character sequencing using kernel method [11].

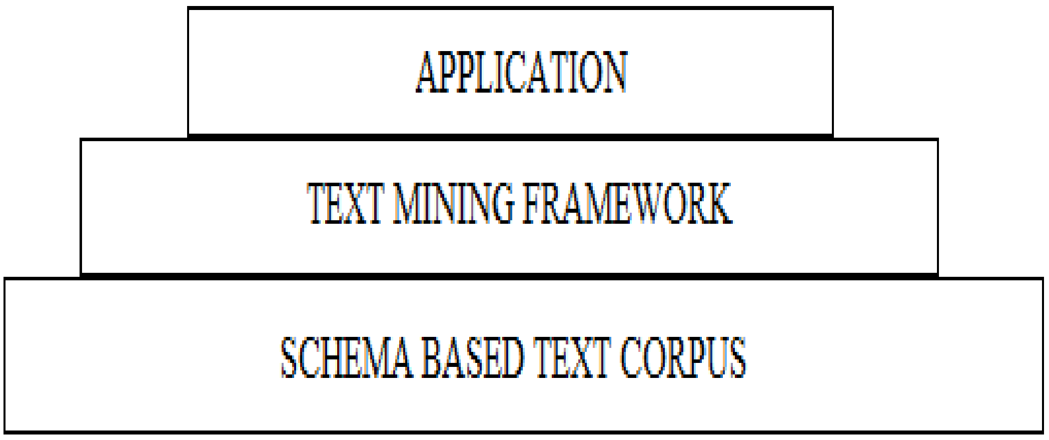


**Fig.4. Framework for proposed Information retrieval system**

A new framework is proposed which uses keyword document matrices and provides interface to access the document corpus as character sequence. The proposed framework contain application layer as the topmost layer, which provides user interface to enter query and get the result.

Text mining framework is used; various text pre-processing algorithms like stop-word list, keyword vocabulary creation, index construction and index compression will be performed in this layer.

Third layer is a Schema Based Text corpus which is text document collection based on user profile.

It is a collection of documents or text database. The elements of Schema Based Text Corpus are text documents holding the actual text corpus and local metadata based on different profiles as discussed in feedback reweigh to keywords section.

Text document collection contains two slots wiz sorting global metadata and database support. Two different metadata are used namely Schema Based Text Corpus metadata and document metadata. Document metadata is for information specific to text document and Schema Based Text Corpus on user profile.

User writes a query which is submitted to the IR engine in the form of Keyword list. User data and Query list is stored in database and is also given to the expert system, based on existing patterns for that particular search for a particular user profile the documents are retrieved and stored in the resultant set. The resultant set is then passed on to the retrieval process and the resultant retrieved relevant documents are passed on to user for interactive feedback. If user is satisfied the search is over otherwise the query is again enriched with new keywords till the users is satisfied with the search or abort the search process.

| **Doc id.** | **Document title & Author** |
|---|---|
| D1 | File Systems and Management, PCP Bhatt/IISc, Bangalore M2/V1/June 04/21 |
| D2 | AWK Tool in Unix, PCP Bhatt/IISc, Bangalore M12/V1/June 04/23 |
| D3 | Search and Sort Tools,P.C.P Bhatt/IISc, Bangalore OS/M11/V1/2005 |
| D4 | Principles of Operating Systems, Prof. Nalini Venkatasubramanian (Venkat) |
| D5 | Operating System Structures<br>Prof. Nalini Venkatasubramanian |
| D6 | Processes and Threads<br>Prof. Nalini Venkatasubramanian |
| D7 | CPU Scheduling<br>Prof. Nalini Venkatasubramanian |
| D8 | Introduction to Operating System, PCP Bhatt/IISc, Bangalore M1/V1/June 04/15 |
| D9 | Memory Management,P.C.P Bhatt/IISc, Bangalore OS/M21/V2/2005 |

**Table 1. Title and morphological information for documents Registered in Database**

## 6. Implementation

The proposed model is implemented using tm package of R tool. The implementation steps are:

### 6.1 Get document Collection and Morphological analysis

The documents are registered, followed by its morphological analysis and Meta data analysis before storing it into database/ corpus. The Corpus is named as Reuter. Morphological analysis is performed on document corpus; Title, Authors, Date of creation, last modification and Description are extracted from the documents. Our Document corpus contains nine documents whose description is given as in table 1, table 2 and table 3.

### 6.2 Metadata analysis

Morphological analysis is followed by metadata analysis. LOM of IEEE Learning Technology and Dublin core Metadata are two metadata. Our implementation is based on R- tool which uses Dublin core metadata.

### 6.3 User profile and user profile manager

User profile manager maintains various registered user data and search history, education level, assessment of knowledge and status of participation in current search session. User registration information helps the expert system to know and provide information about users to the delivery document module.

### 6.4 Similarity analysis

Similarity analysis is performed followed by clustering of keywords and keywords that share similar meaning. Similarity algorithms used is Jacquard followed by correlation matrix calculation and docterm matrix.

### 6.5 Frequency analysis

The numbers of times the set of relevant keywords appear that characterize the document.

**Table1.2. Keywords extracted from documents.**

| Doc id. | Keywords |
|---|---|
| D1 | File system, management, storage, lists, executables, applications, and disk. |
| D2 | Awk, employee, command, CRUD, string, structured set, hourly wage. |
| D3 | grep, test file, Bhatt, test file, sort, int, command |
| D4 | Operating systems, operating systems principles, I/O, hardware, Memory, CPU. |
| D5 | Memory, operating system, I/O, protection, interrupt, storage, management. |
| D6 | Communication, CPU, memory, queue, I/O, kernel thread, PCB. |
| D7 | Scheduling, CPU, queue, jobs, priority, burst, response time. |
| D8 | Computer system, memory support, communication, applications, operating system, instruction. |
| D9 | Memory, references, segment, frame, loader, instruction, second level. |

**Table 3. Description of documents Registered in Database.**

| Doc id. | Description |
|---|---|
| D1 | □ Page 1----------------------Module 2: File Systems and Management<br>□only temporary storage of information.<br>□ Management through a file system.<br>□ That software which allows users and applications to organize their files.<br>□ In the disk, it should know its size, when was it created, i.e. date and time of creation.<br>□ls command: Unix's ls command which lists files and subdirectories in a directory is<br>□ are shareable executable. |
| D2 | □ Page 1----------------------Module 12: AWK Tool in Unix<br>□ Each record may contain fields like employee<br>□ As AWK is used to process a structured set of records, we shall use a small file called.<br>□ Name, employee's hourly wage, and the number of hours the employee has worked.<br>□ Let us use the awk command with input file awk.test as shown below:<br>□ bhatt@falerno [CRUD] =>awk '$3> 0 {print $1, $2 * $3}' awk.test<br>□ Where the options may be like a file input instead of a quoted string. |
| D3 | □Page1---------Search and Sort Tools<br>□grep stands for general regular expression parser.<br>□ int declarations in a program called add.c.<br>□file called test file.<br>□bhatt@SE-0 [F]>>grep '[0-9]' testfile<br>□drwxr-xr-x 2 bhatt b512 Oct 15 13:15 M1<br>□string command to find if there are ascii strings in certain. |
| D4 | □ Principles of Operating Systems Lecture 1<br>□ CPU Scheduling, Process Synchronization<br>□Memory Management, Virtual Memory<br>□ I/O Systems<br>□ Use computer hardware efficiently. |
| D5 | □ Lecture 2 - Operating System Structures.<br>□Storage Structure, Storage Hierarchy<br>□Process Management, Memory Management, Secondary Storage<br>□Management, I/O System Management, File Management, Protection<br>□Interrupt transfers control to the interrupt service routine. |
| D6 | □CPU scheduling information process priority, pointer<br>□of I/O devices allocated<br>□Process (PCB) moves from queue to queue<br>□Memory, ready and waiting to execute.<br>□May have several user threads per kernel thread<br>□process communication and process synchronization. |
| D7 | □ CPU Scheduling |

| | |
|---|---|
| | FCFS, Shortest Job First, Priority, Round Robin,<br>CPU-I/O Burst Cycle<br>Selects which jobs to temporarily suspend/resume to |

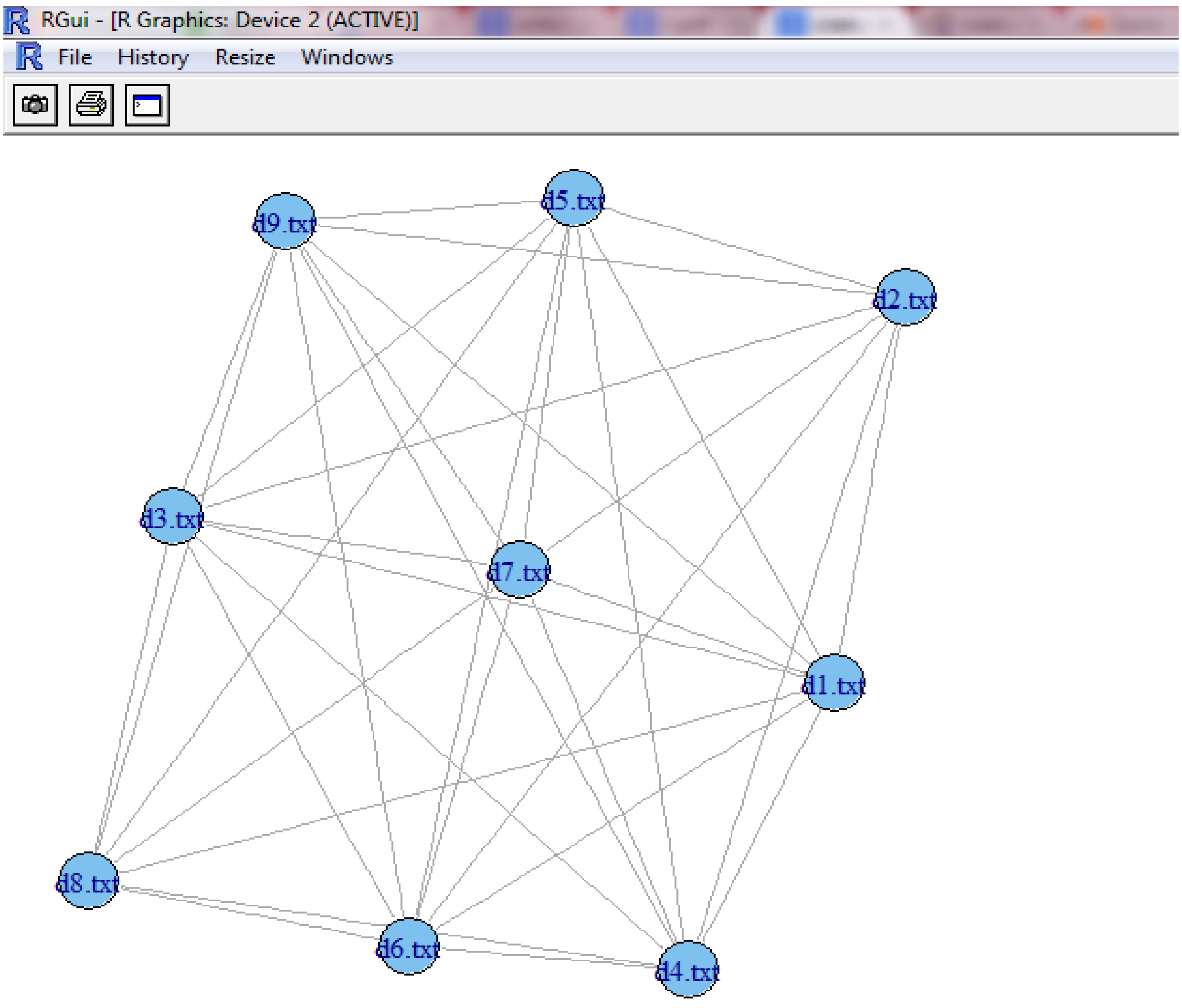
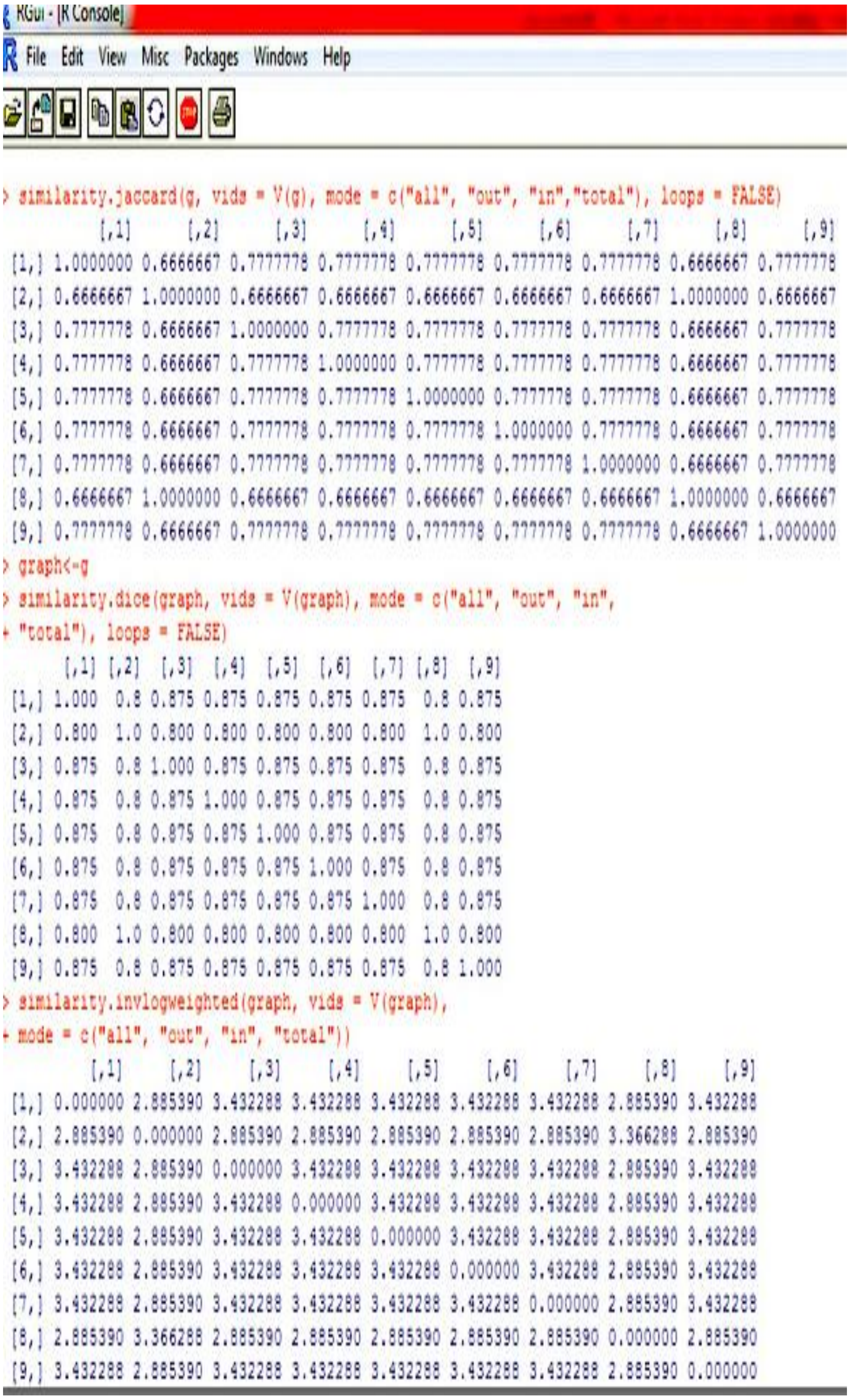
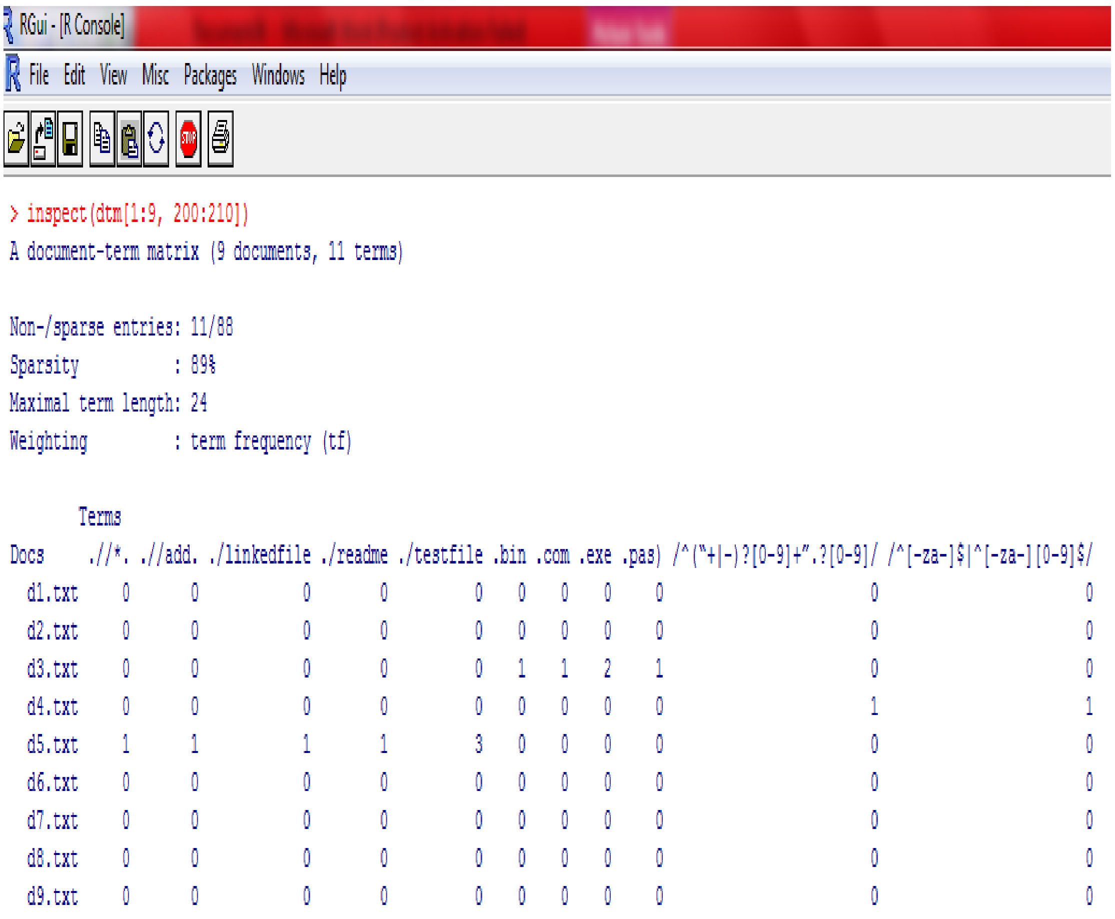


**Fig.5. Graph of documents of corpus**



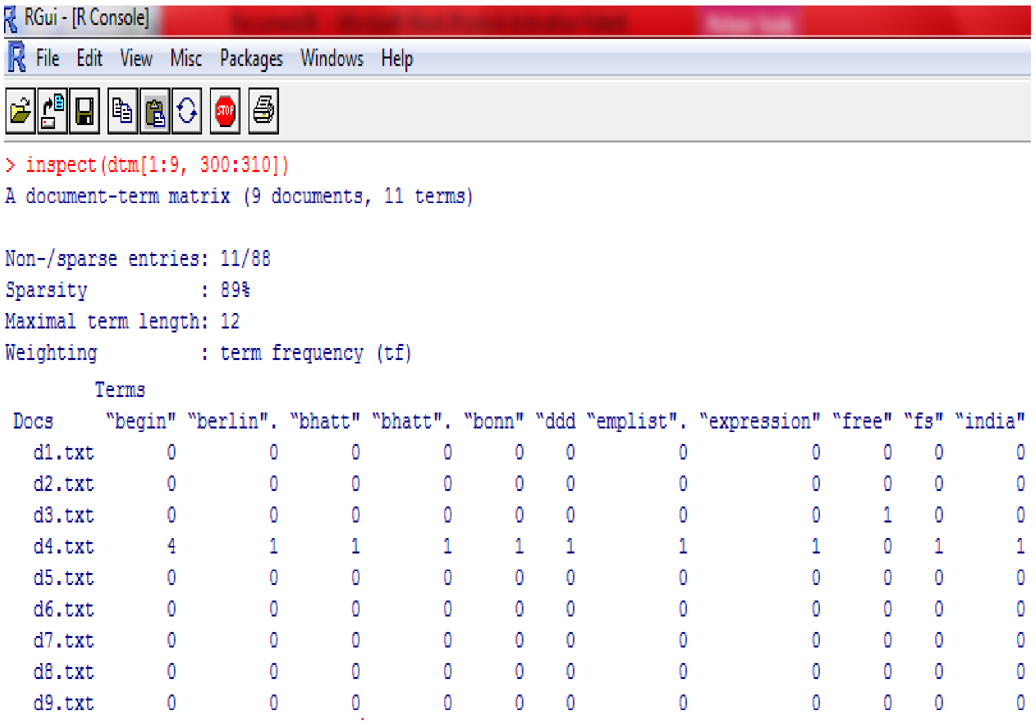
```
> similarity.jaccard(g, vids = V(g), mode = c("all", "out", "in","total"), loops = FALSE)
          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,8]      [,9]
[1,] 1.0000000 0.6666667 0.7777778 0.7777778 0.7777778 0.7777778 0.7777778 0.6666667 0.7777778
[2,] 0.6666667 1.0000000 0.6666667 0.6666667 0.6666667 0.6666667 0.6666667 1.0000000 0.6666667
[3,] 0.7777778 0.6666667 1.0000000 0.7777778 0.7777778 0.7777778 0.7777778 0.6666667 0.7777778
[4,] 0.7777778 0.6666667 0.7777778 1.0000000 0.7777778 0.7777778 0.7777778 0.6666667 0.7777778
[5,] 0.7777778 0.6666667 0.7777778 0.7777778 1.0000000 0.7777778 0.7777778 0.6666667 0.7777778
[6,] 0.7777778 0.6666667 0.7777778 0.7777778 0.7777778 1.0000000 0.7777778 0.6666667 0.7777778
[7,] 0.7777778 0.6666667 0.7777778 0.7777778 0.7777778 0.7777778 1.0000000 0.6666667 0.7777778
[8,] 0.6666667 1.0000000 0.6666667 0.6666667 0.6666667 0.6666667 0.6666667 1.0000000 0.6666667
[9,] 0.7777778 0.6666667 0.7777778 0.7777778 0.7777778 0.7777778 0.7777778 0.6666667 1.0000000
> graph<-g
> similarity.dice(graph, vids = V(graph), mode = c("all", "out", "in",
+ "total"), loops = FALSE)
      [,1] [,2]  [,3]  [,4]  [,5]  [,6]  [,7] [,8]  [,9]
[1,] 1.000  0.8 0.875 0.875 0.875 0.875 0.875  0.8 0.875
[2,] 0.800  1.0 0.800 0.800 0.800 0.800 0.800  1.0 0.800
[3,] 0.875  0.8 1.000 0.875 0.875 0.875 0.875  0.8 0.875
[4,] 0.875  0.8 0.875 1.000 0.875 0.875 0.875  0.8 0.875
[5,] 0.875  0.8 0.875 0.875 1.000 0.875 0.875  0.8 0.875
[6,] 0.875  0.8 0.875 0.875 0.875 1.000 0.875  0.8 0.875
[7,] 0.875  0.8 0.875 0.875 0.875 0.875 1.000  0.8 0.875
[8,] 0.800  1.0 0.800 0.800 0.800 0.800 0.800  1.0 0.800
[9,] 0.875  0.8 0.875 0.875 0.875 0.875 0.875  0.8 1.000
> similarity.invlogweighted(graph, vids = V(graph),
+ mode = c("all", "out", "in", "total"))
         [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]     [,8]     [,9]
[1,] 0.000000 2.885390 3.432288 3.432288 3.432288 3.432288 3.432288 2.885390 3.432288
[2,] 2.885390 0.000000 2.885390 2.885390 2.885390 2.885390 2.885390 3.366288 2.885390
[3,] 3.432288 2.885390 0.000000 3.432288 3.432288 3.432288 3.432288 2.885390 3.432288
[4,] 3.432288 2.885390 3.432288 0.000000 3.432288 3.432288 3.432288 2.885390 3.432288
[5,] 3.432288 2.885390 3.432288 3.432288 0.000000 3.432288 3.432288 2.885390 3.432288
[6,] 3.432288 2.885390 3.432288 3.432288 3.432288 0.000000 3.432288 2.885390 3.432288
[7,] 3.432288 2.885390 3.432288 3.432288 3.432288 3.432288 0.000000 2.885390 3.432288
[8,] 2.885390 3.366288 2.885390 2.885390 2.885390 2.885390 2.885390 0.000000 2.885390
[9,] 3.432288 2.885390 3.432288 3.432288 3.432288 3.432288 3.432288 2.885390 0.000000
```

**Fig. 6. Similarity matrix with three different algorithms: igraph package of R – tool.**



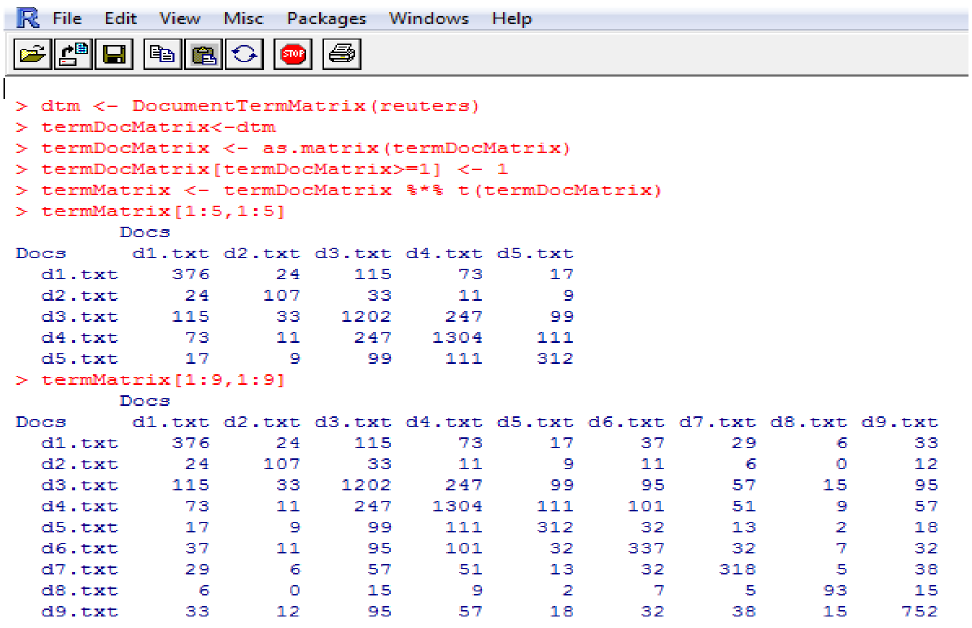
```
> inspect(dtm[1:9, 200:210])
A document-term matrix (9 documents, 11 terms)

Non-/sparse entries: 11/88
Sparsity           : 89%
Maximal term length: 24
Weighting          : term frequency (tf)

        Terms
Docs     .//*. .//add. ./linkedfile ./readme ./testfile .bin .com .exe .pas) /^("+|-)?[0-9]+".?[0-9]/ /^[-za-]$|^[-za-][0-9]$/
  d1.txt     0       0            0        0          0    0    0    0     0                        0                        0
  d2.txt     0       0            0        0          0    0    0    0     0                        0                        0
  d3.txt     0       0            0        0          0    1    1    2     1                        0                        0
  d4.txt     0       0            0        0          0    0    0    0     0                        1                        1
  d5.txt     1       1            1        1          3    0    0    0     0                        0                        0
  d6.txt     0       0            0        0          0    0    0    0     0                        0                        0
  d7.txt     0       0            0        0          0    0    0    0     0                        0                        0
  d8.txt     0       0            0        0          0    0    0    0     0                        0                        0
  d9.txt     0       0            0        0          0    0    0    0     0                        0                        0
```

**Fig.7.a. Document Term matrix**



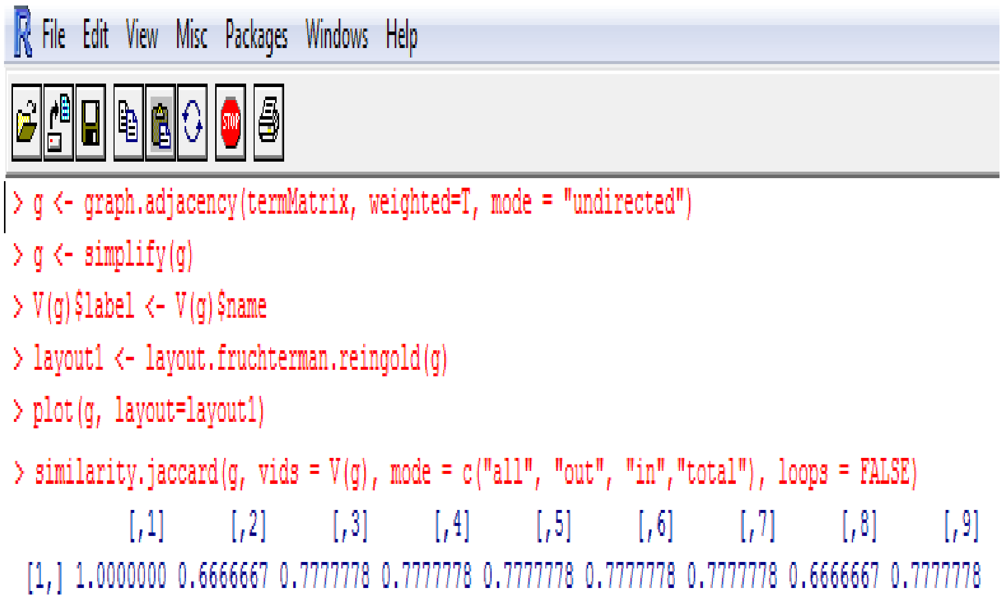
```
> inspect(dtm[1:9, 300:310])
A document-term matrix (9 documents, 11 terms)

Non-/sparse entries: 11/88
Sparsity           : 89%
Maximal term length: 12
Weighting          : term frequency (tf)
            Terms
 Docs     "begin" "berlin". "bhatt" "bhatt". "bonn" "ddd "emplist". "expression" "free" "fs" "india"
   d1.txt       0         0       0        0      0    0          0            0      0    0       0
   d2.txt       0         0       0        0      0    0          0            0      0    0       0
   d3.txt       0         0       0        0      0    0          0            0      1    0       0
   d4.txt       4         1       1        1      1    1          1            1      0    1       1
   d5.txt       0         0       0        0      0    0          0            0      0    0       0
   d6.txt       0         0       0        0      0    0          0            0      0    0       0
   d7.txt       0         0       0        0      0    0          0            0      0    0       0
   d8.txt       0         0       0        0      0    0          0            0      0    0       0
   d9.txt       0         0       0        0      0    0          0            0      0    0       0
```

**Fig.7.b. Document Term matrix**



```
> dtm <- DocumentTermMatrix(reuters)
> termDocMatrix<-dtm
> termDocMatrix <- as.matrix(termDocMatrix)
> termDocMatrix[termDocMatrix>=1] <- 1
> termMatrix <- termDocMatrix %*% t(termDocMatrix)
> termMatrix[1:5,1:5]
          Docs
Docs       d1.txt d2.txt d3.txt d4.txt d5.txt
  d1.txt     376     24    115     73     17
  d2.txt      24    107     33     11      9
  d3.txt     115     33   1202    247     99
  d4.txt      73     11    247   1304    111
  d5.txt      17      9     99    111    312
> termMatrix[1:9,1:9]
          Docs
Docs       d1.txt d2.txt d3.txt d4.txt d5.txt d6.txt d7.txt d8.txt d9.txt
  d1.txt     376     24    115     73     17     37     29      6     33
  d2.txt      24    107     33     11      9     11      6      0     12
  d3.txt     115     33   1202    247     99     95     57     15     95
  d4.txt      73     11    247   1304    111    101     51      9     57
  d5.txt      17      9     99    111    312     32     13      2     18
  d6.txt      37     11     95    101     32    337     32      7     32
  d7.txt      29      6     57     51     13     32    318      5     38
  d8.txt       6      0     15      9      2      7      5     93     15
  d9.txt      33     12     95     57     18     32     38     15    752
```

**Fig.7.c. Document Term matrix**



```
> g <- graph.adjacency(termMatrix, weighted=T, mode = "undirected")
> g <- simplify(g)
> V(g)$label <- V(g)$name
> layout1 <- layout.fruchterman.reingold(g)
> plot(g, layout=layout1)

> similarity.jaccard(g, vids = V(g), mode = c("all", "out", "in","total"), loops = FALSE)
          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,8]      [,9]
 [1,] 1.0000000 0.6666667 0.7777778 0.7777778 0.7777778 0.7777778 0.7777778 0.6666667 0.7777778
```

**Fig.8. Correlation matrix of documents**

## 7. Conclusion

We introduced and proposed a new architecture and framework for Information retrieval system. Proposed Information retrieval system uses User profile, relevance feedback and expert system to enhance the information retrieval process. The limitation with the existing system is that, as user feedback is considered so the system can become user biased.

## 8. REFERENCES

[1] D.R. Green and S.D. King, “Internet-Based Information Systems: The Forth Estuary Forum (FEF) System”, D.R. Green and S.D. King (eds.), Coastal and Marine Geo-Information Systems, 451–465, 2003 Kluwer Academic Publishers. Printed in the Netherlands.

[2] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, “Modern Information Retrieval” Addison-Wesley Longman Publishing co. 1999.

[3] http://sifaka.cs.uiuc.edu/czhai/pub/ir-tok.pdf

[4] Christopher D. Manning,Prabhakar Raghavan,Hinrich Schütze, “An Introduction toInformationRetrieval” Cambridge University Press Cambridge, England

[5] M. F. Porter, “An algorithm for suffix stripping”, published in \Program\, \14\ no. 3, and pp 130-137, July 1980.

[6] http://nlp.stanford.edu/IR-book/html/htmledition/a-first-take-at-building-an-inverted-index-1.html

[7] http://nlp.stanford.edu/IR-book/html/htmledition/index-compression-1.html

[8] http://cogsys.imm.dtu.dk/thor/projects/multimedia/textmining/node5.html

[9] Lee, D. L., Chuang, Huei and Seamons, “Document ranking and the vector-space model”. IEEE Software, Vol.14 (2), p. 67-75, 1997.

[10] Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze,”Introduction to Information Retrieval”.Cambridge University Press. 2008.

[11] Dmitry Zelenko, Chinatsu Aone, Anthony Richardella, “Kernel Methods for Relation Extraction”, Journal of Machine Learning Research 3 (2003) 1083-1106, SRA International USA.