ISSN: 2321-8169
68 - 73

An Approach for Design Search Engine Architecture for Document Summarization

Bijoy Kumar Mandal[1], Rajesh Mukherjee[1], Payel Majumder[1], Supravat Mondal[1], Arindam Biswas[1]
Computer Science & Engineering, ECE
NSHM Knowledge Campus Durgapur
Durgapur, India
writetobijoy@gmail.com

Dr. A K Bandyopadhyay[2]
Electronics & Communication Engineering
NIT Durgapur
Durgapur, India
akbece12@yahoo.com

***Abstract*—** Query focused multi document summarization is an emerging area of research. A lot of work has already been done on the subject and a lot more is going on. The following document outlines the effort done by us in this particular field. This work proposes an approach to address automatic Multi Document text summarization in response to a query given by a user. For the explosion of information in the World Wide Web, this work proposed a new method of query-focused multi-documents summarization using genetic algorithm, search engine are used to extract relevant documents and genetic algorithm is used to extract the sentences to form a summary, and it is based on a fitness function formed by three factors: query-focused feature, importance feature, and non-redundancy feature. Experimental result shows that the proposed summarization method can improve the performance of summary, genetic algorithm is efficient. We have developed a very powerful search engine one. On the same note, it also has a great potential for growth. It can be easily applied for systems with not only a few documents but for very large systems with a large number of documents
.

Keywords-component**;** *DUC, Multi Document Summarization, GA, Semantic, Query, Score, Term Frequency*

*****

I. INTRODUCTION

Normally, when users need some information on a particular query, they fire up a search engine and search for the required thing [1]. But, this approach has one big problem. Opening and analyzing each and every web page to get the required information becomes very tedious and time consuming [2]. Different documents deal with different aspects of the given query. Thus, it becomes very difficult for the user to analyze and decide which data and up to what extent is usable for them. There are different types of queries but more importantly when a user enters a search string, he/she wants the relevant information [3, 4]. The search engine looks up all the documents from the database and generates long results. That is the query result in its simplified form. Search engines like Google, Bing etc. generate the query result in this way.

In our context, a query can be thought of as a word or a set of words entered by the user. In our system, the intention of the user is to view a paragraph level summary about the search term. In stricter, i.e. technical terms that we have used here, a query is the string entered by the user from which stop-words (like forms of be, articles, prepositions, etc.) have been removed. The terms thus, obtained give us the perspective in which the user wishes to search. The main aim of query focused multi-document summarization is to extract a meaningful summary of a given search query from the documents available in its database [4]. A more plausible solution to the above problem would be to provide the users with a single document in response to their queries. The system would generate a simplified document that would contain more or less of all the aspects of the given query from its pool of pre-loaded documents [5, 6]. This system will not only allow the users to view all the information that they need in a single place, but would also allow them to easily analyzes and digest the information. This also enables the users to decide whether they need to see a particular perspective of their query in greater detail. But here we need to emphasize that aspects of a single query is very important. Often we see that we look for a certain aspect in a given query, however the result which we get is not relevant. So it is the most important part that each & every aspect, disregarding of its importance in the context of the given query should be there in multi-document summarization. This will not only enhance the usefulness of multi-document summarization, but it can serve a large variety of users performing query with the same exact string, however they are looking for different topical aspect.

Let us take the example of a query and assume that the user is simply searching for “Java”. A search engine will normally return results which direct the users to various documents dealing with the different aspects of “Java”. Some will point to basics like class, data structures, etc. The others might point to documents with advanced topics. Still others might point to free and paid tutorials. The users might get overwhelmed by these results. They might even lose their perspective because of so many aspects which again lead to other aspects. So a plausible solution is to collect the documents and generate a summary [7]. Now, the user does not have to scrounge for information as it is readily available. If required, they can even zoom in some particular aspect of the query like “Advanced Java” which will lead them to even finer results. Thus in simple terms, what is being done here is to generate an index like outcome which can further lead to the narrowing down of search results. Now take this example in a bit different way, a java programmer as well as an experienced java architect can search for the string ‘Advanced Java’. However the needs of both the persons are different, so we need to highlight the necessary topical aspects so that both the persons can get what they were looking for. The expert might be looking for a particular example, say, how to use the java plug-in for different browser, while the naïve programmer might be searching for the topics & the free tutorial which falls in the category of ‘Advanced Java’. So the document contains all the aspects which satiate the need of different user. This will enhance the productivity of the multi-document summarization.

II. TECHNOLOGICAL CHALLENGES

The multi-document summarization task has turned out to be much more complex than summarizing a single document [8], even a very large one. This difficulty arises from inevitable thematic diversity within a large set of documents. A good summarization technology aims to combine the main themes with completeness, readability, and conciseness. Document Understanding Conferences (DUC), conducted annually by NIST, have developed sophisticated evaluation criteria for techniques accepting the multi-document summarization challenge [9, 10]. An ideal multi-document summarization system does not simply shorten the source texts but presents information organized around the key aspects to represent a wider diversity of views on the topic. When such quality is achieved, an automatic multi-document summary is perceived more like an overview of a given topic [11]. The latter implies that such text compilations should also meet other basic requirements for an overview text compiled by a human.

- Text within sections is divided into meaningful paragraphs
- Gradual transition from more general to more specific thematic aspects
- Good readability

The latter point deserves additional note - special care is taken in order to ensure that the automatic overview shows:

- No paper-unrelated "information noise" from the respective documents (e.g. web pages)
- No dangling references to what is not mentioned or explained in the overview
- No text breaks across a sentence
- No semantic redundancy.

III. COMMERCIAL USES OF MDS

The multi-document summarization technology is now coming of age a view supported by a choice of advanced web-based systems that are currently available.

A. Ultimate Research Assistant

The Ultimate Research Assistant performs text mining on Internet search results to help summarize and organize them and make it easier for the user to perform online research. Specific text mining techniques used by the tool include concept extraction, text summarization, hierarchical concept clustering (e.g., automated taxonomy generation), and various visualization techniques, including tag clouds and mind maps. To use this tool, the user types in the name of a topic and the tool will search the web for highly relevant resources, and organize the search results into a rich, easy-to-understand research report.

B. iResearch Reporter

Commercial Text Extraction and Text Summarization system, free demo site accepts user-entered query, passes it on to Google search engine, retrieves multiple relevant documents, produces categorized, easily-readable natural language summary reports covering multiple documents in retrieved set, all extracts linked to original documents on the Web, post-processing, entity extraction, event and relationship extraction, text extraction, extract clustering, linguistic analysis, multi-document, full text, natural language processing, categorization rules, clustering, linguistic analysis, text summary construction tool set.

C. News Blaster

It is a system that helps users find the news that is of the most interest to them. The system automatically collects, clusters, categorizes, and summarizes news from several sites on the web (CNN, Reuters, Fox News, etc.) on a daily basis, and it provides users a user-friendly interface to browse the results.

D. News-In-Essence

It may be used to retrieve and summarize a cluster of articles from the web. It can start from a URL and retrieve documents that are similar, or it can retrieve documents that match a given set of keywords. News-In-Essence also downloads hundreds of news articles daily and produces news clusters from them.

E. News Feed Researcher

It is a news portal performing continuous automatic summarization of documents initially clustered by the news aggregators (e.g., Google News). News Feed Researcher is backed by the free online engine covering major events related to business, technology, U.S. and international news. This tool is also available in the on-demand mode allowing a user to build a summary on any selected topic.

F. Scrape

This is like a search engine, but instead of providing links to the most relevant websites based on a query, it scrapes the pertinent information off of the relevant websites and provides the user with a consolidated multi-document summary, along with dictionary definitions, images, and videos.

G. Jist Web

Jist Web is an Efficient Query Specific Multiple Document Summarizer that was developed by Jasta.

.

IV. MATHEMATICAL EXPRESSIONS

The main reason behind every perfect completion of a project is to realize the problem properly. As this topic of query focused multi-document summarization is new and one of the rising research topic in India, we have to go through several international conference proceedings and several international journals from many recognized organizations. Obviously, it takes much time to study the problem. After studying the problem, we have generated the mathematical expression of the problem.

- Degree of importance of i-th sentence of d-th document:

$$I_{d,D}(S_i) = \frac{W_{d,D}(t)}{w_i} \quad \text{……………………(i)}$$

Here, $W_{d,D}(t)$: the weight of the term, t of d-th document
w_i : no.of words present in i-th sentence.

- Degree of importance of j-th attribute of d-th document:

$$I_{d,D}(A_j) = \sum_{k=1}^{m} \frac{W_{d,D}(t)}{w_k} \quad \text{………………..(ii)}$$

Here, $W_{d,D}(t)$: the weight of the term, t of d-th document
m: no. of sentences present in the j-th attribute of d-th document.

Our main goal is to maximize the value of $\sum_{j=1}^{4} I_{n,D}(A_j)$as there are 4 attribute (except doc_no) in the doc_info table, j ranges from 1 to 4. n represents no. of documents go with the given query after searching operation performed. This is our fitness function.

The fitness function is $\sum_{j=1}^{4} I_{n,D}(A_j)$ and it is a maximizing function. Now, in case of finding the maximum value of this function, we have to develop a (nX4) 2- dimensional integer array where n is no. of documents go with the given query after searching operation performed and 4 is the no. of attribute that describe a document in the doc_info table in the database. The array structure is given below:

Doc no. = 1	[	$I_{I,D}(A_1)$	$I_{I,D}(A_2)$
	$I_{I,D}(A_3)$	$I_{I,D}(A_4)$	]
Doc no. = 2	[	$I_{2,D}(A_1)$	$I_{2,D}(A_2)$
	$I_{2,D}(A_3)$	$I_{2,D}(A_4)$	]
…………………………………………………………………			
Doc no. = n	[	$I_{n,D}(A_1)$	$I_{n,D}(A_2)$
	$I_{n,D}(A_3)$	$I_{n,D}(A_4)$	]

Now, the parameters used in GA are population size, cross-over operator and no. of iterations. We are representing a solution in a form of a document. So, at the beginning, the populations are also the no. of documents that match with the given term. We are using cross over operator for generating new off-springs that has better fitness value compare to its parents. Here, we are using 2-point cross-over operator for generating new off-springs, so there are 3 possible ways that cross-over can happen between two parents. The cross-over rate is 0.6

Parent 1 (best fitness)	$I_{I,D}(A_1)$	r1 |	$I_{I,D}(A_2)$	$I_{I,D}(A_3)$	r2 |	$I_{I,D}(A_4)$
Parent 2 (worst fitness)	$I_{2,D}(A_1)$	|	$I_{2,D}(A_2)$	$I_{2,D}(A_3)$	|	$I_{2,D}(A_4)$
Offspring 1	$I_{I,D}(A_1)$		$I_{I,D}(A_2)$	$I_{I,D}(A_3)$		$I_{I,D}(A_4)$
Offspring 2	$I_{2,D}(A_1)$		$I_{2,D}(A_2)$	$I_{2,D}(A_3)$		$I_{2,D}(A_4)$

In the above, cross-over between two parents (one is having best fitness value & other is having worst fitness value) is shown using 2 crossing over point. Position of crossing over point is also generated randomly. r1 and r2 are random numbers between 1 and 3 (as there are 4 attributes, so no. of points where crossing over can take place is 3), simultaneously, they must satisfy 1<r1<r2< 3.

As we are choosing parents with one have best fitness and other have worst fitness, so after crossing over, we are always getting two off-springs which have better fitness than the parent with worst fitness. As a result the parent with worst fitness is replaced with his next generation (a new offspring with better fitness). Thus, after iteration by iteration, new populations are generated and also we are going towards our goal.

V. ALGORITHM

The first thing after a user gives the query, we have to find the proper term by removing all the stop words using stops_word table and keeping just the words that are important. After that we have to search for the documents in the database where we can find the term. For that searching purpose, we have designed search_doc table where we can check the term with the keyword attribute of that table and we can easily find which doc_no holds the description of that given term. Then we can store the doc_no in a 1-dimentional (1-D) integer array. That array is holding the multiple documents for that given query. After all the iterations are completed, we are sorting the populations according to their fitness values and finally we get the optimum document as a solution. Now, for extracting summary from that optimized document, we are taking 2-3 sentences from each attribute of that document using degree of importance of a sentence and club them. That is our ultimate optimized query-focused multi-document summary. The algorithm for multiple text summarizations is follow as

Step 1 : Search query is given.
Step 2 : Remove stop[words (if any) and take the main word(term).
Step 3: Go to the database and check with the keyboard attribute of search_doc table with the term. Whenever a keyword matches with the term, store its doc_no to a 1-Dimentional integer array.
Step 4: The 1-D array holds the document number related to the term. After that go to the doc_info table in the database and search the full documents of those doc_no. In file table there are multiple rows (multiple documents) and multiple columns (for describing different parts of that term like introduction, description, features, conclusion).
Step 5: Now, find term frequency (*tf*) for each document (for each row) and inverse document frequency (*idf)* for calculating weight of the term in that documents.
Step 6: After that find degree of importance of each sentence, $I_d(S)$ and find the degree of importance of each attribute in the document (row), $I_d(A)$.
Step 7: Continue step 5 and 6 for all the documents of doc_no that stored previously in a 1-d array.
Step 8: In case of summarization and to develop the coding, we store all the $I_d(A)$ in 2-D array (m X n) where m is document number (values that are already store the 1-D array) and n is the attribute number.
Step 9: We perform Genetic Algorithm (GA) with its cross-over operation with that 2-D array and fitness function is the

summation of all the $I_d(A)$ of a single row of that array. We use GA to find the maximum fitness.
Step 10: After finding the maximum possible fitness, we could easily find the optimized document (solution) from those multiple documents and from that optimized document, we take one or two sentences from each attribute using $I_d(S)$.
Step 11: Ultimately we club those sentences of each attribute to produce the best summarization.

VI. IMPLEMENTATION & EXPERIMENTAL RESULTS

We have developed a web based enterprise application for multi-document summarization problem. There are only two types of user. One is common people and other is the admin himself.

- **Admin:** They can also use the system to find a summary but the main difference between common users and the admin is they can insert documents into the database by logging in to the website.

- **Common Users:** They use the system to find a suitable summary by giving a search query.

A. *Search Space*

In solving problems, some solution will be the best among others. The space of all feasible solutions (among which the desired solution resides) is called search space (also called state space).

- Each point in the search space represents one possible solution.
- Each possible solution can be "marked" by its value (or fitness) for the problem.
- The GA looks for the best solution among a number of possible solutions represented by one point in the search space.
- Looking for a solution is then equal to looking for some extreme value (minimum or maximum) in the search space.
- At times the search space may be well defined, but usually only a few points in the search space are known.

Using GA, the process of finding solutions generates other points (possible solutions) as evolution proceeds.

B. *Hardware Requirements*

- Server side
  - ➢ Processor : Intel(R) CORE i3-2310M CPU @ 2.10 GHz
  - ➢ RAM : 1 GB
  - ➢ Disk Space : 20 GB
- Client side
  - ➢ Processor: Pentium IV
  - ➢ RAM: 128 MB
  - ➢ Disk Space: 500 MB

C. *Software Requirements*

- Server side
  - ➢ Operating System : Windows 7 Server Edition
  - ➢ Java Development Kit (JDK) version 6u27
  - ➢ Eclipse IDE for JEE development (Front End)
  - ➢ Database : MySQL (Back End)
  - ➢ Apache tomcat application server version 6.0.16

- Client side
  - ➢ Operating System : Windows XP
  - ➢ Java Development Kit (JDK)
  - ➢ Web Browser

D. *Database Design*

The next step is to design the database for the MDS problem. Designing the database for multi document summarization problem is the most important task, because the documents, from which summary is to be extracted, are reside in the database. There are three tables in the database as Search Document (search_doc) table, Document Information (doc_info) table and Stops Word (stops_word) table as shown in Table I, Table II & Table II respectively.

Table I Document Information

Attribute Name	Data Type	Size	Constraint	Reference
doc_no	INT	10	Primary Key	
Title	VARCHAR	100	Not Null	
Part 1	VARCHAR	2000	Not Null	
Part 2	VARCHAR	2000	Not Null	
Part 3	VARCHAR	2000	Not Null	
Part 4	VARCHAR	2000	Not Null	

Table II Search Document

Attribute Name	Data Type	Size	Constraint	Reference
doc_no	INT	10	Foreign Key	doc_info
keyword	VARCHAR	30	Not Null	

Table III Stop Word

Attribute Name	Data Type	Size	Constraint	Reference
words	VARCHAR	30	Not Null	

The main intention of creating the database is to insert documents. Here, we are entering a single document in 4 parts. There are reasons behind it. First of all it helps to create modularity, secondly we can easily extract sentences from each part to generate summary. When we insert a document, there should be high cohesion and low coupling between sentences as well as between four parts. There is a separate text space for document keywords (user has the option of entering multiple keywords for a single document) in the GUI where user can add a document into the database. Document no. is generated automatically by the system. The stops_word table helps us to remove stop words like a, an, and, or, the, of i.e. mainly articles and prepositions from the user given search query.

E. *Flow Diagram*

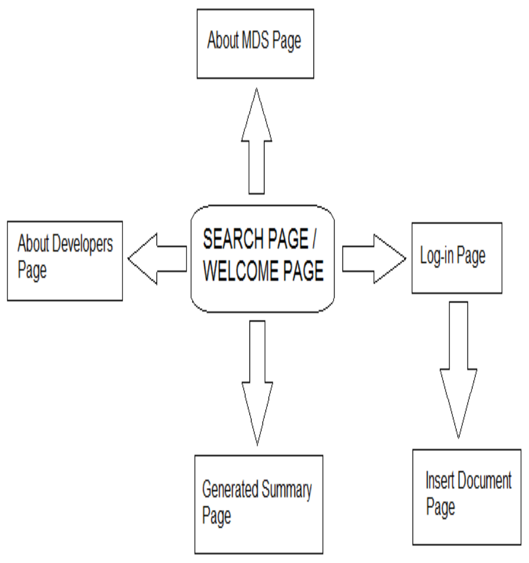
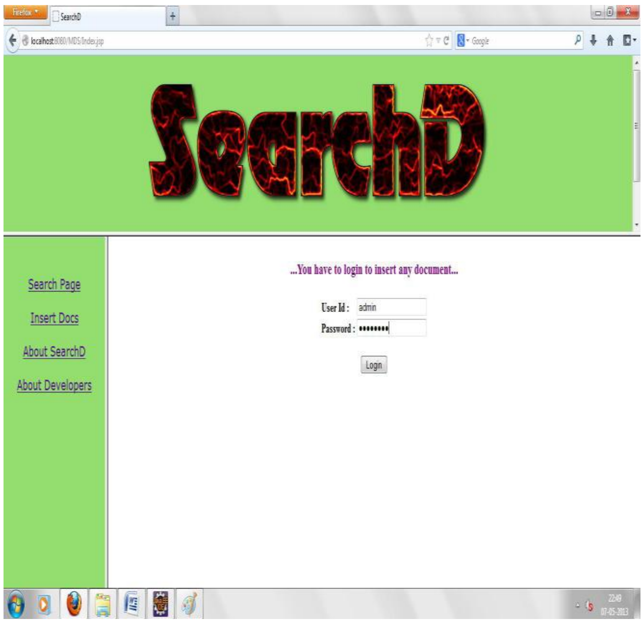


Figure 1. Flow Diagram

F. *Screen Shots*

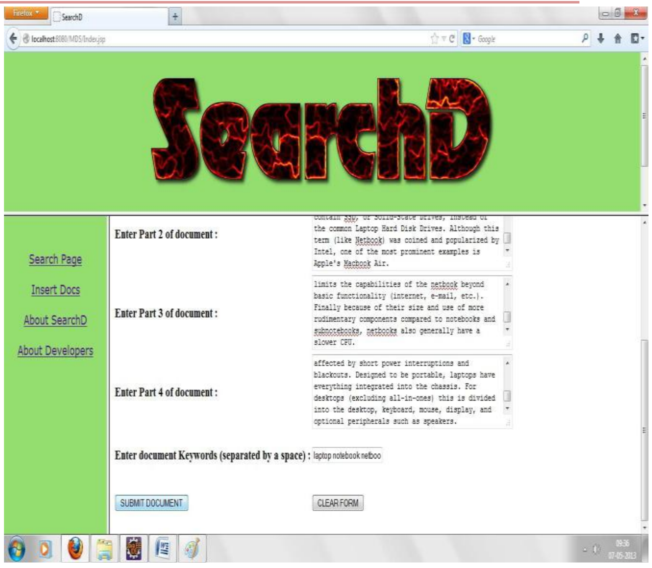


...You have to login to insert any document...

User Id : admin
Password : ••••••••

G.

Figure 2. Loing Page



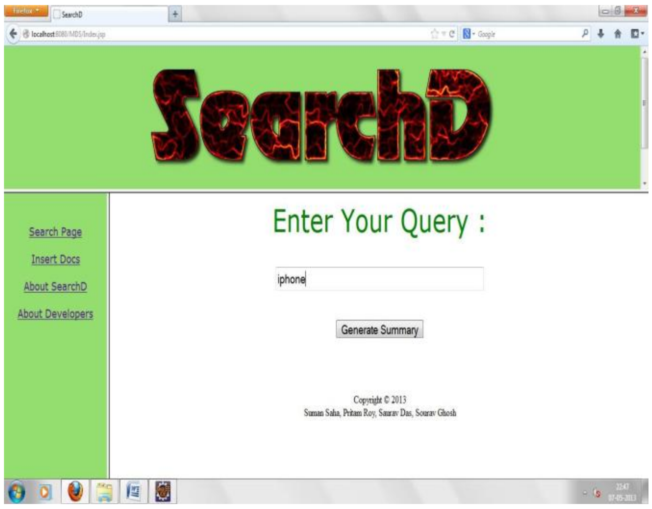
Enter Part 2 of document : contain SSD, or Solid-State Drives, instead of the common Laptop Hard Disk Drives. Although this term (like Netbook) was coined and popularized by Intel, one of the most prominent examples is Apple's Macbook Air.

Enter Part 3 of document : limits the capabilities of the netbook beyond basic functionality (internet, e-mail, etc.). Finally because of their size and use of more rudimentary components compared to notebooks and subnotebooks, netbooks also generally have a slower CPU.

Enter Part 4 of document : affected by short power interruptions and blackouts. Designed to be portable, laptops have everything integrated into the chassis. For desktops (excluding all-in-ones) this is divided into the desktop, keyboard, mouse, display, and optional peripherals such as speakers.

Enter document Keywords (separated by a space) : laptop notebook netboo

Figure 3. Documents Insertation Page

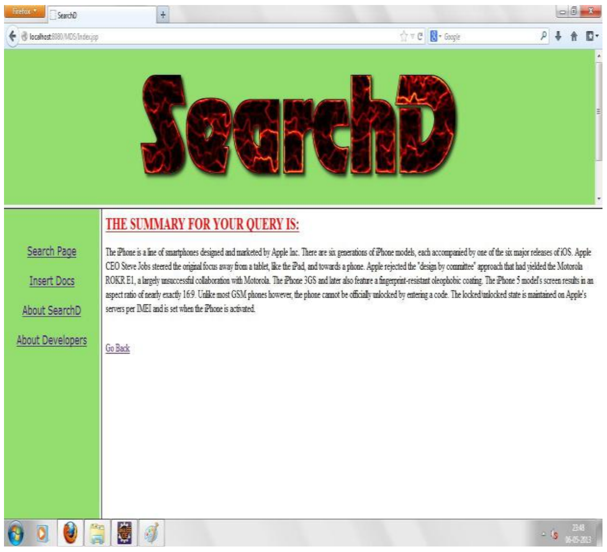


Enter Your Query :

iphone

Copyright © 2013
Suman Saha, Pritam Roy, Saurav Das, Sourav Ghosh

Figure 4. Searching Page



THE SUMMARY FOR YOUR QUERY IS:

The iPhone is a line of smartphones designed and marketed by Apple Inc. There are six generations of iPhone models, each accompanied by one of the six major releases of iOS. Apple CEO Steve Jobs steered the original focus away from a tablet, like the iPad, and towards a phone. Apple rejected the "design by committee" approach that had yielded the Motorola ROKR E1, a largely unsuccessful collaboration with Motorola. The iPhone 3GS and later also feature a fingerprint-resistant oleophobic coating. The iPhone 5 model's screen results in an aspect ratio of nearly exactly 16:9. Unlike most GSM phones however, the phone cannot be officially unlocked by entering a code. The locked/unlocked state is maintained on Apple's servers per IMEI and is set when the iPhone is activated.

Figure 5. Final Summary Page

VII. CONCLUSION

In this report, we have discussed query focused multi-document summarization problem and it is solving procedure using a soft-computing technique – Genetic Algorithm (GA). We are actually not familiar with informative summaries (output is a summary of given input query) rather than we are much more comfortable with indicative summaries (output is information of document which helps the user to decide whether the document should be read or not). But informative summaries are much more significant than indicative summaries. Here comes the importance of our project. It successfully generates indicative summaries that can be used by people to pin-point what they want when they search for a particular term on the web. The modern web is a very complex structure of all kinds of data. Summaries generated by our application program will help the users to fan-out all the unnecessary things while putting forward the details.

REFERENCES

[1] R.Kowsalya , R.Priya and P.Nithiya – "Multi Document Extractive Summarization Based On Word Sequences" , IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 2, March 2011 ISSN (Online): 1694-0814.

[2] Toshihiko Sakurai ; Akira Utsumi – "Query-based Multidocument Summarization for Information Retrieval" , Proceedings of NTCIR-4, Tokyo, April 2003 - June 2004.

[3] Lacatusu, A. Hickl, K. Roberts, Y. Shi, J. Bensley, B. Rink, P. Wang, and L. Taylor. 2006. Lcc"s gistexter at duc 2006: "Multi-strategy multi-document summarization", In Proceedings of DUC"06, 2006.

[4] Surabhi Gupta and Ani Nenkova and Dan Jurafsky – "Measuring Importance and Query Relevance in Topic-focused Multi-document Summarization" , Stanford University Stanford, CA 94305.

[5] Aliguliyev, R. M. "Automatic Document Summarization by Sentence Extraction" in Journal of Computational Technologies, pp.5–15, 2007.

[6] Chao Shen, Tao Li – "Learning to Rank for Query-focused Multi-Document Summarization" , 11th IEEE International Conference on Data Mining 2011 IEEE DOI 10.1109/ICDM.2011.91 1550-4786/11

[7] Tingting He ; Fang Li ; Zhuomin Gui ; Jinguang Chen – "Query-Focused Multi-document Summarization Using Keyword Extraction" , IEEE International Conference on Computer Science and Software Engineering, pp 20-23, 2008.

[8] "Genetic Algorithms in Search, Optimization & Machine Learning" by David E. Goldberg – Addition-Wiley Press, Pearson Education Publication ISBN 978-81-7758-829-3

[9] L. Vanderwende, H. Suzuki, and C. Brockett. Microsoft research at duc 2006: Task-focused summarization with sentence simplification and lexical expansion, In Proceedings of DUC"06, 2006.

[10] Ramiz M. Aliguliyev , "A New Sentence Similarity Measure And Sentence Based Extractive Technique For Automatic Text Summarization", Expert Systems with Applications36 pp.7764–7772, 2009.

[11] C. Long, M. Huang, X. Zhu, and M. Li, "Multi-document summarization by information distance," in Proceedings of the 9th IEEE International Conference on Data Mining. IEEE, pp. 866–871, 2009.

[12] C. Shen and T. Li, "Multi-document summarization via the minimum dominating set," in Proceedings of the 23rd International Conference on Computational Linguistics. Association for Computational Linguistics, pp. 984–992, 2010.