CHAPTER 6
AUTOMATIC MULTIPLE DOCUMENT TEXT SUMMARIZER

6.1 Introduction
The number of documents and users on the World Wide Web (WWW) are increasing rapidly. This increases the size of any repository of search system. This increase in size has lead to information overload problem. The search system like Google provides a large number of URLs corresponding to the search keywords. The results retuned by the Search Engine (SE) contain a brief description of the text; which is limited to at most three lines of the text. Moreover, these lines are the initial lines of the document, which may or may not provide some meaningful information to the user. That’s why Automatic Text Summarization (ATS) techniques are used. This helps the end user in understanding the main ideas of documents quickly (Sornil and Gree-Ut 2006; Jung et al.2005). The generation of a small summary from the contents of a webpage would be helpful for the users to get an instantaneous idea about the contents of the webpage without actually going through the entire contents. The user can have a concise overview in a short time. This can greatly enhance the retrieval efficiency.

This chapter proposes an ATS, which can summarize the selected document(s) by preserving the original meaning of the document. It considers several issues like reducing each document up to some extent, ordering of the sentences coming from different sources by keeping the logical and grammatical structure in proper format while implementing the proposed summarizer. ATS systems are designed to take a single as well as cluster of articles as input and generates a brief and fluent summary of the most important information. To implement an effective auto summarizer, authors utilize the concept of DOM tree to analyze the content of webpage and HTML Agility pack for the creation of HTML DOM
tree. In recent years, the summarization applications were developed for news, webpages, specialized remedial information, scientific articles etc. These systems seem inadequate when they are used.

6.2 Problems with Traditional and Existing ATS

With the current explosion of available information, it becomes very difficult to know about everything, even in a narrow field, and it is necessary to rely on finding the information when it is needed. This problem occurs in all domains, regardless of the type of information sought. One solution to this problem is offered by the information processing and management field. This field tries to facilitate the access to the information either through better retrieval algorithms and filtering, or by reducing the information to simpler forms that can be processed more easily.

In general, webpage summarization derives from text summarization techniques creates a challenge to summarize the webpages automatically and efficiently. All this happens because of the diverse structure and contents of the webpages. These contents of webpage may be in the form of text, table, bullets, images and links etc. Furthermore, webpage may contain diversified subjects and information content, which is required to be consider while summarizing the webpage.

The idea of comparing the machine generated summary to human-written summaries seems highly believable because human behavior introduces inconsistency. Several experimental studies have shown that the degree of agreement between several human written summaries is low. Humans generated summaries do not share the same views as to what the content of the ideal summary should consist of and moreover humans do not always remain faithful to the content selection (Yew and Hovy 2002). That’s why many ATS techniques have been proposed in the literatures (Kupiec et al. 1995; Lin and Hovy 1997; Barzilay and Elhadad 1997; Conroy and Leary 2001; Radev 2004;
Evans et al. 2005). The major concern of all these techniques is primarily related to text summarization with effective representation of results. But all these techniques still suffer from many problems as given below:

(i) They used preprocessed data which diminishes the importance of their proposed method.

(ii) They provide preprocessed summary to the user. But the dynamic nature of the Web obsoletes such types of preprocessed summary. So, what is the benefit of downloading and creating a summary of all the searched results? It results in wastages of network bandwidth and system processing also.

(iii) They used lesser number of tags while cleaning and summarizing the webpage. As a webpage contains several types of information and tags, so, this may leads to loss of important information.

(iv) Traditional summarization techniques (Kupiec et al. 1995; Lin and Hovy 1997; Barzilay and Elhadad 1997; Conroy and Leary 2001) measured the importance of sentence by its location only. But today, such techniques cannot be adopted in ever changing environment.

(v) Traditional text summarizers lack the efficiency due to data redundancy, which provides irrelevant data to end user.

To overcome these problems, an automated framework for summarizing the search results is proposed in next section.

6.3 Proposed Automatic Multiple Document Text Summarizer

To address the problems of existing ATS stated in section 6.2, an Automatic Multiple Document Text Summarizer (Proposed ATS) is presented in this section. The architecture of proposed ATS is shown in Figure 6.1.
The main components of the architecture are Search Engine Interface (SEI), Search Engine Results (SER), Selected URLs (SURLs), Downloaded webpages, webpage Filtration, DOM tree, TB and CB generator, and Summarizer. Each of these components is discussed in detail below:

**Search Engine Interface (SEI):** This module is the heart of the whole system through which user can interact with the proposed summarizer. When a user gives a query on the interface of the SE, then SE provides a list of URLs to the end user. The returned results of the SE are stored temporarily. These results are
removed automatically as soon as a new request for summary generation is arrived or system is restarted.

**Selected URLs (SURLs):** SE returns a long list of URLs corresponding to the given query. But summary is generated for few of them. So, user have to select required URLs to be downloaded. These are only the webpages which are used while summarization. Therefore, SURLs contains selected and downloaded webpages which are selected by the user by using SEI.

**Downloaded Webpages Repository:** It contains the downloaded webpages corresponding to the selected URLs in local repository of SE.

**Webpage Filter:** An intermediate representation of the code is generated after the pre-processing of text. One of the pre-processing stages is eliminating the stop words or empty words from the text. There is a set of empty words in every language; common to all domains which are easily identified. Normally a single webpage contains a good number of functional words which are commonly used to join other words together to form the sentence structure understandable. When considered as single tokens, these words do not provide any semantic benefit towards the context of the text. Such words include articles like a\an\the, prepositions such as on\at\in\since\ago\before\past\till\until\by etc. and conjunctions such as and\or\not\for\yet\so etc. It follows that a list of such functional words are referred to as the stop word, and these can be removed from text without losing the essence of the content. This process is known as removal of stopping words. Problem arises when determining which words should be included in the list because some words might be important for some topics that’s why the stopping word list should be decided first. In fact, it is also assumed that a word which appears in at least 80% of the webpage of a particular collection is useless for purpose of information retrieval. These words are considered empty and normally are removed.
**DOM Tree, TB and CB Generator:** The Document Object Model (DOM) is an Application Programming Interface (API) for HTML webpage. It defines the logical structure of webpage and the way a webpage is accessed and manipulated. With the DOM any one can build webpage, navigate their structure, add and modify the content. The DOM tree is generated corresponding to the filtered webpage. Authors of this thesis used it for generating two types of blocks i.e. Topic Block and Concept Block, which are explained as follows:

- **Topic Block (TB):** The leaf nodes of DOM tree are considered as Micro Blocks (MB). The MB of the same parent tag forms a TB. Therefore, TB contains the contents of the webpage.

- **Concept Block (CB):** The TB having the similar information is merged to form a CB. The concept based similarities are measured by considering the given query keywords, feature keywords, frequency, location of the sentence, tag in which the text appears in the webpage and uppercase words etc. The CBs are generated from the contents of concept based segmentation. The TB having the similar information is merged to form a CB. The process of concept based segmentation is described with the help of a tree structure as shown in Figure 6.2. This tree structure uses the concept of DOM tree in which the rectangular nodes represent the HTML tags. These tags can be used to some higher level topics or sub-topics. The circular nodes represent the information content within the tag. The semantic string of the contents is maintained in circular nodes from left to right (Chen and Shen 2009). The DOM trees created corresponding to the user selected URLs are processed to generate the summarized result. Leaf nodes are considered as MB which are the basic building blocks of the summary.

The contribution of each term of the main topic is also examined. This model of concept based mining tries to captures the semantic structure of each term within a sentence rather than the frequency of the term alone. This method of similarity
Figure 6.2: Concept Based Segmentation
measurement leads to the other similarity measurement technique that is based on term analysis models (Sornil and Ut 2006). This system determines the words which contribute more and then labels them by a semantic role labeler. The semantic role labeler is able to identify the verb argument structures of each sentence in the TB.

Total number of label generation depends on the amount of information in the sentence. A single sentence may have many labels so it may include many verbs associated with their arguments. Generally, the word contributing more to the meaning of the sentence appears more number of times in the sentence. Hence these words are comparatively higher in frequency.

The CB formation is completed offline for selected and downloaded webpages in the repository. The CBs id, conceptual terms, frequency and list of sequence numbers of sentences of each of these CBs are stored in a local database, which are required for processing at run time. These CBs contain related information which are scattered throughout the webpage. The set of sentences in each of these CBs are actually present in different parts of the webpage. Every type of data required and collected during processing is automatically erased whenever system is instructed for new comparison or restarted. A new database is created every time when a comparison is to be made.

Thus, it saves a lot of memory space required for storing the information. This memory space can be utilized for other purposes. Before actually performing Concept Based Segmentation it is essential to extract and filter the contents of all the internal links present in the user selected URL because the summary generation involves the contents not only from the home page but from the internal links present in the URL as well. Therefore, the contents of all the internal links are extracted and then filtered. The DOM tree is generated for all the internal links in the way discussed above.
**Summarizer Module:** It is responsible for generation of summary. To generate the summary CBs are compared based on the selection of feature keywords (f). The f is used to generate the summary of the webpage(s). The relevancy of CB to the f is measured by using the equation (6.1).

\[
\text{sim}_f (f, CB_i) = \frac{\sum_{t \in f \cap CB_i} C \cdot tf_{ti}}{\sqrt{\sum_{t \in CB_i} (C \cdot tf_{ti})^2}} \quad \text{... (6.1)}
\]

Where

- f: set of feature keywords, f={f_1, f_2, ..., f_n}
- CB_i: Concept Block i, for which the list of concept terms and their frequency were already identified.
- Sim_f (f, CB_i): Similarity between feature keyword string f and Concept Block CB_i
- t: set of common terms between f and CB_i
- Ctf_{ti}: frequency of term t in CB_i

Sim_f(f, CB_i) is measured using the term frequency of CB. The CB having maximum number of matching conceptual terms gets higher score. Sim_f(f, CB_i) assign Similarity Score (SS) between 0 and 1, and the similarity increases as SS increases. The CB with maximum SS is considered as the superset of the summary to be generated. The significance of each sentence is measured with respect to the query string. The sentences are considered in the descending order of their SS. According to the number of sentences required, the sentences are extracted from these CBs and comparative summary is generated.

**6.3.1 Working of Proposed Summarizer**

The proposed architecture of ATS is explained in this section and the work flow for the same is shown in Figure 6.3 where user gives the query at SEI.
Figure 6.3: Work Flow of Proposed Summarizer
Then Query Processor of SE matches the query terms with the indexed database, retrieves the results, ranks them and presents them to the end user. Now, based on the snippets retrieved, user selects single/multiple URL(s) for summarization. The webpages corresponding to the selected URLs are downloaded and stored in local repository. These webpages are cleaned by removing unwanted HTML tags (like META tag, ALIGN tag, etc.) which do not contribute much more for further processing. Then DOM tree for cleaned webpage are created by Agility tool. DOM tree is used to identify the MB, TB and CB. The CBs are created and are provided to the WordNet. WordNet is a large lexical database of English language which is freely available on the Internet. It resembles a vocabulary, which groups the words together based on their meanings. It compares each TB with other TB and based on the comparison it generates a SS. A threshold value of SS for summary generation is decided priory. The sentences having the SS equal to or above the predetermined threshold value are considered in summary generation. Moreover, if webpage contains tabular data, then the destination webpage creates its own table and puts the data in it from the source webpage without any alteration.

It must be noted that destination table must contains same number of rows and columns as are contained in source table. The format of outer border can be different in both tables but data remain same. This complete working of concept based segmentation is represented with the help of an algorithm shown in Figure 6.4.
Algorithm: Concept Based Segmentation

Input: Webpages(di)

Begin
Step 1: Mark all leaf nodes as individual MB in the DOM tree
Step 2: //Generate TB from MB
   TB = {tb1, tb2, …, tbn} // where tbi={si1, si2, ..sin1}, si⊂ tbi,
Step 3: //Calculate the similarity of TBs.

\[ Sim_f (f, C B_i) = \frac{\sum_{t \in f \cap C B_i} C t f_{ti}}{\sqrt{\sum_{t \in C B_i} (C t f_{ti})^2}} \]

Step 4: //Merge the TBs having similarity score above the threshold
CB={set of TBs}
Step 5: Return CBs

End

Output: Set of CB {Cb1 ,..Cbn} of di.

Figure 6.4: Algorithm for Concept Based Segmentation

6.3.2 Experimental Setup

The proposed framework was implemented in ASP.NET. Apart from this authors also used HTML agility tool, WordNet and Nugget as supporting software’s for the proper functioning of the proposed ATS. HTML Agility packs was used for the creation of HTML DOM tree. NUGET was used for the
installation of HTML Agility pack that builds a DOM tree. It was a .NET binary code library that allows parsing of the webpages. Further, WordNet was used for expressing a distinct concept of a webpage. It contains English verb, noun, adjective and adverbs organized on the basis of their word meanings. Some words can represent several meanings and same meanings can be represented by various forms. So, it compares each TB with other TBs and assigns a SS.

The proposed ATS was tested on various webpages of different websites and in different time slots. But here for discussion author select only two of them i.e. www.msit.in and www.piet.edu. Both of these websites were related to engineering colleges located in New Delhi and Panipat respectively. These websites were tested on the featured keyword called “Placement”. The webpages corresponding to the featured keyword were downloaded and cleaned by removing the stopping words (discussed in section 6.3). One list of such types of words is shown in Table 6.1. The webpage has been filtered by removing the unwanted HTML tags also. These tags are meta, align and CSS style tags. Furthermore, ‘&nbsp’ has been replaced by space characters as these characters do not contribute to summary generation. The filtered document helps in the formation of TB.

The list of words and their respective frequency of occurrence for these TBs were identified and their similarity was calculated using Similarity Generator formula (discussed in section 6.3). The TB having similarity value 0.5 and above were combined to form the CB.
<table>
<thead>
<tr>
<th>Alphabet</th>
<th>Stop words</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a, about, again, all, almost, also, although, always, among, an, and, any, are, as, at</td>
</tr>
<tr>
<td>b</td>
<td>be, because, been, before, being, both, but, by</td>
</tr>
<tr>
<td>c</td>
<td>can, could</td>
</tr>
<tr>
<td>d</td>
<td>did, do, does, done, due, during</td>
</tr>
<tr>
<td>e</td>
<td>each, either, enough,</td>
</tr>
<tr>
<td>f</td>
<td>for, found, from, further</td>
</tr>
<tr>
<td>h</td>
<td>had, has, have, having, here, how, however</td>
</tr>
<tr>
<td>i</td>
<td>i, if, in, into, is, it, its, itself</td>
</tr>
<tr>
<td>j</td>
<td>just</td>
</tr>
<tr>
<td>k</td>
<td>kg, km</td>
</tr>
<tr>
<td>m</td>
<td>made, mainly, make, may, mg, might, ml, mm, most, mostly, must</td>
</tr>
<tr>
<td>n</td>
<td>nearly, neither, no, nor</td>
</tr>
<tr>
<td>o</td>
<td>obtained, of, often, on, our, overall</td>
</tr>
<tr>
<td>p</td>
<td>perhaps</td>
</tr>
<tr>
<td>q</td>
<td>quite</td>
</tr>
<tr>
<td>r</td>
<td>rather, really,</td>
</tr>
<tr>
<td>s</td>
<td>should, show, showed, shown, shows, since, so, some, such</td>
</tr>
<tr>
<td>t</td>
<td>than, that, the, their, theirs, them, then, there, therefore, these, they, this, those, through, thus, to</td>
</tr>
<tr>
<td>u</td>
<td>upon, use, used, using</td>
</tr>
<tr>
<td>v</td>
<td>very</td>
</tr>
<tr>
<td>w</td>
<td>was, we, were, what, when, which, while, with, would</td>
</tr>
</tbody>
</table>
6.3.3 Results and Discussion

The webpages corresponding to the feature keyword “Placement” were downloaded and cleaned. DOM tree was generated correspondingly. The leaf nodes were marked as MB and TB. Then similarity score of sentences was calculated and the sentences scored 0.5 and above were considered for summarization. Moreover, the SS near about the threshold value are assumed to be closer to the requirement of the user. So the links having SS between >0.47 and <0.5 are also returned to the user. Now it’s upon the user whether to select the recommended URLs or not. The obtained summarized results are shown in Figure 6.5 where the summarized result shows the parallel comparison of both the selected webpages. The achieved results contained textual data for normal description.

Moreover, summarized results also contain tabular data coming from the selected webpages. This tabular data contains the information from designated webpages and put it into its own table. There was no manipulation applied on tabular data. The data of this table was just picked from the source document and was placed in destination table. From this multi-document summarized result, based on featured keyword any one can easily compare these colleges and can reach to meaningful conclusion.

The results of proposed ATS is compared with some online summarizers (see Table 6.2) i.e. Free Summarizer, Auto Summarizer and Tool4noobs. The comparison was also made with two other summarizers i.e. MEAD and Comparative Summarizer (available in literatures). All these summarizers were compared on 13 parameters viz. Method used for Summary Generation, Tabular Data used, Techniques used, URL/Textual Input, Percentage of Text Compaction, Single/Multi Document, User Control, Way of Presentation, Satisfaction, Usage, Best/Recommended words, Retrieval Time and Database.
Figure 6.5: Summarized Results of Selected Webpages

Most of the results of Table 6.2 indicate that the proposed summarizer is superior than the existing summarizers available online as well as in literatures. Therefore, by the proposed system of text summarization, the searching and analyzing time of the user is reduced significantly.
Table 6.2: Comparison of Different Text Summarizers

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Method used for summary Generation</td>
<td>Extractive</td>
<td>Extractive</td>
<td>Extractive</td>
<td>Extractive</td>
<td>Extractive</td>
<td>Extractive</td>
</tr>
<tr>
<td>Tabular data used</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>NO</td>
<td>NO</td>
<td>yes</td>
</tr>
<tr>
<td>Techniques used</td>
<td>Frequency of characters</td>
<td>Frequency of characters</td>
<td>Frequency of characters</td>
<td>Lex rank, Centroid Position</td>
<td>DOM tree, Concept based segmentation</td>
<td>DOM tree, Concept Based segmentation</td>
</tr>
<tr>
<td>URL/Textual input</td>
<td>Textual</td>
<td>Textual</td>
<td>URL/Textual</td>
<td>Text Documents</td>
<td>Webpages</td>
<td>Webpages</td>
</tr>
<tr>
<td>% of text compaction (to user)</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>NO</td>
</tr>
<tr>
<td>User control</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Less Control</td>
<td>More Control</td>
<td>More Control</td>
</tr>
<tr>
<td>Satisfaction</td>
<td>No</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium to high</td>
<td>Higher</td>
<td></td>
</tr>
<tr>
<td>Way of presentation</td>
<td>Small text box</td>
<td>Small text box</td>
<td>Small text box</td>
<td>Sufficient text box</td>
<td>Sufficient text box</td>
<td></td>
</tr>
<tr>
<td>Usage</td>
<td>Short summary</td>
<td>Short summary</td>
<td>Short summary</td>
<td>Short summary</td>
<td>Sufficient Comparative Summary is generated</td>
<td>Sufficient Comparative Summary is generated</td>
</tr>
<tr>
<td>Best/Recommended words</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Retrieval time</td>
<td>Less time</td>
<td>Less time</td>
<td>Less time</td>
<td>Tolerable</td>
<td>Tolerable</td>
<td>Tolerable</td>
</tr>
<tr>
<td>Database</td>
<td>Dynamic</td>
<td>Dynamic</td>
<td>Dynamic</td>
<td>Static Database</td>
<td>Static Database</td>
<td>Dynamic Database</td>
</tr>
</tbody>
</table>
The above discussion and comparison with other summarizers shows that proposed ATS not only provides the summary of the text but comparison also which makes it different from other summarizers. Reading the entire article, dissecting it and separating the important ideas from the raw text, takes more time and effort. Reading an article of 550 words can take at least 15 minutes. But proposed ATS can summarize texts of 550-5000 words in a fraction of a second. This allows the user to read less data but still receive the most important information and make solid conclusions which in turn improves productivity. Furthermore, better interface and way of summarized results presentation makes it more valuable. In proposed ATS, database is created at run time only and stored in local data base for some time. This database gets vanished as soon as system is restarted or a new search is made by the user. This dynamic nature makes it better than others. As the proposed ATS downloads only those webpages which are instructed by the user only. So less number of webpages are downloaded which reduces lot of traffic on the Internet.

6.4 Applications of the Proposed Automatic Text Summarizer

This section describes the application areas for the proposed ATS.

(i) First application of the proposed ATS is in the area of mass media. Today a news article is written by a journalist, but when typesetting the newspaper the article is shortened manually to the appropriate size so that it can fit in the layout, in between the advertisements. An experiment was described in (Dalianis et al. 2003) where both manual editors and the SweSum text summarizer (Dalianis 2000) were given the task to summarize 334 news texts for the newspaper Sydsvenska Dagbladet. The manually generated summary and automatic summarized text were compared and it was found that both of the texts were almost identical. The same experiment was carried out for SMS format (maximum 160 characters) and the results from SweSum were considered suitable to be
used directly in newspaper production. So, proposed ATS can directly be used in News and SMS also because it can reduce the text up to desired limit.

(ii) Business Intelligence systems or news monitoring systems are very common for flow of news. This news flow can be summarized in such a way that user can obtain an overview of the text before deciding if they should click on the news summary and read the complete news article (McKeown and Radev 1995; McKeown et al. 2003). So the proposed ATS can be used for live applications which can take several news articles and summarizes it to one single news.

(iii) Further one might need a multi-document ATS, in which one could use MEAD (Radev et al. 2004). The area of medicine and biomedicine where several attempts are made to use ATS for different user groups such as patients, physicians, nurses and scientists. The (Hirst et al. 1997) medical digital libraries produce user adapted information towards individual patient specific needs and general health education. But proposed ATS provides textual description containing the information which was adapted to the user interest. It can provide the information for medical, technical and educational domain etc. from a single interface.

(iv) The application area of ATS was to create abstracts. In 1959, due to storage constraints it was not possible to store the whole document together with the title and author name (Luhn 1959). Today there is a wide range of application areas for ATS, the most common and obvious one is in information retrieval. It can be observed from the resultant list of SE where a summarized part of each retrieved document is presented together with the search terms of the user which is referred to as snippets. But such types of summaries are limited to 2 or 3 lines only which may or may not describe the content inside the webpage. So the propose ATS can be used to overcome such type of limitations.
6.5 Summary

In this chapter problems with the traditional and existing automatic text summarizers were identified such as diversified subject and information contents of webpage, irrelevancy of preprocessed summarization, wastage of network bandwidth and system processing, use of lesser HTML tags in summary generation, summary generation by location of sentences and redundancy of results. To address these problems the authors of the thesis proposed an architecture called “Automatic Multiple Document Text Summarizer”. The main features of the proposed ATS are selection of URLs, downloading of required webpages only, filtration of webpages, generation of DOM tree, selection of webpages based on similarity only, recommendation of URLs and user friendly SE interface.

The proposed ATS was implemented in ASP.Net. Apart from this, authors of the thesis also used HTML agility tool, WordNet and Nugget software as a supporting software. The performance of proposed ATS was compared with three online summarizer viz. Free Summarizer, Auto Summarizer and Tools4noobs. The comparison was also made with two other summarizers i.e. MEAD and Comparative Summarizer. They were compared on the following thirteen parameters: Method used for Summary Generation, Tabular Data used, Techniques used, URL/Textual Input, Percentage of Text Compaction, Single/Multi Document, User Control, Way of Presentation, Satisfaction, Usage, Best/Recommended words, Retrieval Time and Database.

Experimentally, it was found that the proposed ATS reduced the load on the network by downloading only require number of webpages. Proposed ATS considered more number of HTML tags than the existing summarizers. Tabular data were also presented to the user if available in the selected webpage. Text could be compressed up to desired limit by adjusting the threshold value. The proposed summarizer could summarize single as well as multiple webpages and could provide the comparison in parallel. Furthermore, user control over the
summarizer and satisfaction while searching and summarizing was found better than the existing summarizers. Moreover, it could also recommend the best suitable links to the user. Its retrieval time was slightly more than the existing summarizer which could be attributed to the similarity calculation of the webpage dynamically.