Chapter 3

Automatic English Text Summarization

Automatic text summarization is the technique, where a computer is endorsed with the work of summarizing a text. A text is entered into the computer and a summarized text is returned, which is a non-redundant extract from the original text. According to Radev et al, a summary can be loosely defined as a text that is produced from one or more texts, that conveys important information in the original text (s), and that is no longer than half of the original text(s) and usually significantly less than that. In other words, the main goal of a summary is to present the main ideas in a document in less space [69].

Automatic text summarization can be used:

- To summarize news to SMS or WAP-format for mobile phones/PDA
- To let a computer synthetically summarize text, when the written text is too long and boring.
- In search engines to present compressed descriptions of the search results.
- In keyword directed subscription of news which are summarized and pushed to the user.
- To search in foreign languages and obtain a translated summary of automatically summarized text.
Isn’t ‘Automatic Summarization’, the need of the era? Yes, ‘Summarization’ -- the art of extracting key content from one or more information sources – is becoming an integral part of our day to day life. People keep abreast of world affairs by listening to news bites. They take investment decisions based on the stock market updates. They go to movies largely on the basis of reviews. With summaries, they can make effective decisions in less time [5]. A simple reason to prepare a summary would be to gain access to and control the flood of information, as everyone needs to know in brief what is worth reading and what is useful for a particular purpose. Nobody wants to waste time reading what is useless. Summaries save readers’ time by giving an overview or outline of the content. As we can see, many Digital Libraries contain huge amount of information and a simple search would yield thousands of pages of content, which is to be submitted to the user in a brief manner. In the same manner, if at all, a user wants to know the content in some ‘Research Papers’ offered in some website, it would be extremely tiresome to go through all the pages and then find out that the information is irrelevant to you, where a brief ‘summary’ of the entire thesis can be presented, which helps a lot. The authors predict that the summarization tools will play an important role in conquering the vast information universes ahead.

We must stress, this is not a new phenomena – it began even before the invention of printing. Scholars in the early ‘Middle Ages’ found it necessary to compile summaries of what was known. These were the
first encyclopedias. With this long history, it is somewhat surprising that the process of summarization has itself been so neglected. Only relatively recently, with access to computers capable of dealing with large textual databases, has there been serious or substantial research in this field. And, Automatic summarization aims at producing a concise, condensed representation of the key information content in an information source for a particular user and task. Interest in automatic summarization continues to grow, motivated by the explosion of on-line information sources and advances in natural language processing and information retrieval. In fact, some form of automatic summarization may be indispensable given the massive information the future universes would be holding in the next century.

Summarization is also regarded as one of the techniques in text mining. The purpose of summarization is to describe the content of a document while reducing the amount of text a user must read. The main ideas of most documents can be described with as little as 20 percent of the original text. Little is lost by summarizing. Most techniques use morphological analysis of words to identify the most frequently used terms - while eliminating words that carry little meaning, such as the articles “a/an/the”. Some algorithms weigh terms used in opening or closing sentences more heavily than other terms, while some approaches look for key phrases that identify important sentences such as “in conclusion”, “most importantly” etc.
Automatic Summarization also plays an important role in Digital Libraries. As there are many books related to a particular subject, the user wants to read the most apt one’s only. Summarization provides the users the resources to read the gist of the book/document (Summary/Abstract). It also helps him to get rid of going through the redundant information, by way of duplicate detection and deletion. Summarization also helps in quick searching. It is optimal to search for the key words in the Summary/Abstract rather than searching in the whole text because the Summary is supposed to contain all the important key words.

While the field continues to progress, there are also many problems that need to be addressed before the promises of automatic text summarization can be fully realized. In addition to developing better theoretical foundations and improved characterization of summarization problems, increased work on evaluation methods and summarization resources, is of great interest.

Incidentally, the authors perceived many writers to be having a common inclination in describing their texts. Most of the writers were observed to be having a tendency to use longer sentences, whenever they are at the crux of explaining a critical theme, perhaps due to an inability to elucidate the complex feature in simpler sentences. It is also observed that the words that are considered more important are observed to have been repeating more frequently than those that are considered less important. Suppose, the writer of the text is explaining
the ‘black hole phenomenon’, he naturally tends to use the word ‘black
hole’, ‘universe’ etc., and many a time. Having knowledge of these
observations, a novel approach is taken in this thesis for 'automatic
summarization' with the following postulates:

- The longest sentence in the text file will be an important sentence in
  the text.
- The frequently repeated words in the text are certainly the important
  words in the text.
- The association of the frequently repeated words with long sentences
  is a key factor.

With these three factors in mind, a well-defined procedure is followed to
prepare a ‘summary’ of the given text. Any duplicate sentences
generated, if at all, are removed by a Similarity-Search algorithm, and
the resultant summary is then compared with the following, for the
purpose of calculating efficiency:

- Summary generated by two jurors, who went through the whole text
  and prepared their own summary, extracting only the original
  sentences from the source text.
- Summary generated over the net by the 'Digital Library of India’s
  (DLI) Text Summarization Utility [97].

And it is concluded that the procedure not only gave good summaries,
but also is efficient in summarizing ‘technical’ content and the
performance is satisfactory in ‘general’ content also.
3.1 Related Work

Automated text summarization has drawn a lot of interest in the natural language processing and information retrieval communities in the recent years. A series of workshops on automatic text summarization [38], special topic sessions in the Association for Computer Linguistics [54], and Special Interest Group on Information Retrieval [46] and Japan [68] have advanced the technology and produced a couple of experimental online systems [5]. Despite these efforts, however, there are no common, convenient, and repeatable evaluation methods that can be easily applied to support system development and just-in-time comparison among different summarization methods.

The problem of automatic summarization poses a variety of tough challenges in both NL understanding and generation. A spate of recent papers and tutorials on this subject, at conferences such as the European Chapter of ACL [5], it is also appropriate to mention about the Recall-Oriented Understudy for Gisting Evaluation [55] here. A brief description of 'ROUGE': Following the recent adoption by the machine translation community of automatic evaluation using the BLEU/NIST scoring process, an in-depth study of a similar idea for evaluating summaries is conducted. The results show that automatic evaluation using unigram co-occurrences, i.e. ROUGE, between summary pairs correlates surprisingly well with human evaluations, based on various
statistical metrics; while direct application of the BLEU evaluation procedure does not always give good results.

Most research on summary generation techniques still relies on extraction of important sentences from the original document to form a summary. There are several methods for measuring the importance of a sentence. Some algorithms calculate a weight for each sentence, taking into account the position of the sentence and word frequencies, while other algorithms use semantic information (e.g. WordNet), in order to find the hierarchy of concepts. In some techniques, Sentences are ranked for potential inclusion in the summary using a weighted combination of statistical, linguistic and typographic features like the position, format and type of sentence, and the word frequency.

Although many summarizing tools are available, it is becoming very difficult to generate meaningful and timely summaries, with the increasing volume of online information. Researchers are investigating for more effective & innovative tools and methods that automatically extract or summary content from information sources. While much has to be done in this area to improve the efficiency, scalability and applicability of these techniques with the growing versatility of the present-day WWW and Internet, the procedure followed by the authors here seem to be an honest attempt to contribute to the effort of making good ‘summaries’, keeping in view the vast benefits that are accrued to many applications like Digital Libraries and others. While it is obvious that the need for a ‘utopian’ summarization utility is more for Digital
Libraries, any attempt in this direction would be welcomed with a grand reception.

### 3.2 Methodology

As a lot of time is saved by reading ‘summaries’ instead of long texts, the savings become invaluable in case of Digital Libraries which contain enormous amounts of information where a simple search should produce a brief summary of the contents to the user. In the same manner, when a user wishes to know the content of some ‘Research Paper’ found in some archive, it is very helpful if we can push a brief ‘summary’ of the entire thesis. It is very apt to say here that the following procedure succeeded in generating summaries which are 15-20% of the input text, without altering the original sentences.

#### 3.2.1 Key Steps in the Procedure

1. First of all, the entire file is scanned and all stop words such as ‘a, the, is, for, in, it’ etc., are removed. For this purpose, a separate file for all these stop words is maintained.

2. Then, the frequency repetition of all unique words is calculated, which we call it as Unique Word Frequency (UWF) and these unique words are then sorted in decreasing order of their frequency. A cut-off mark is chosen by the user as to how many of these most frequent words (KEYWORDS) are to be given priority. We call it ‘Cut-Off Count of KEYWORDS’ (COCK). We will weigh only these KEYWORDS and ignore the rest all words.
3. Similarly, all the sentence lengths are sorted, i.e., the number of words in each sentence. The user is again given the choice to opt for a cut-off mark as to what minimum length of sentence is to be considered for inclusion in the summary. We call this Cut-Off Length of Sentence (COLS).

4. Now, every sentence is weighed as follows:

- If the sentence is longer than the ‘COLS’, it will be included in the summary.
- If the sentence is having equal or more words than the ‘COCK’, it will be included in the summary.

5. All the original sentences that stood qualified through the entire process are given as ‘summary’.

6. If at all any sentence appears twice or any two sentences with similar set of KEYWORDS are extracted into the ‘summary’, a Similarity-Search Algorithm removes such sentences the final ‘summary’ is given to the user.

7. If the user wishes to change the summary, he may opt to go back to step no:2 and repeat the entire process with new ‘cut-off marks’ to get a fresh summary of the contents.
In the above process, the user is given the choice to refresh his summary with a new pair of cut-off marks, if at all he wishes to look at a different summary. In any case, the system has its own default values.

Let us see how figure 3.1 depicts the process of generating summaries, in an algorithmic way:

Procedure generate_summary (Input)
{

File temp = remove_stopwords (Input);

generate_UWF (temp);

select_top10%_of_UWF_as_KEYWORDS ();

COCK = take_choice_from_user ();  // KEYWORD cut-off
sort_all_sentences_by_length ();

COLS = take_choice_from_user ();

summary += validate_each_sentence (COLS, COCK);

//validate if it is longer than COLS --OR--

// having more than ‘COCK’ keywords in it

project_summary ();

if (chosen_by_user) call generate_summary (Input);


_____________________________________________________________________

Fig.3.2 Algorithm Depicting the Entire Process.

As it is evident from the above code that the user is given complete freedom to opt for his own levels of summarization in extracting a summary that gives him the best understanding of the given content. Even if he is not in a position to offer his options, the system has its own default values on which the ‘EXPERIMENTATION’ section below throws some light.

3.2.2 Example
Polymorphism (from the Greek, meaning ‘many forms’) is a feature that allows one interface to be used for a general class of actions. The specific action is determined by the exact nature of the situation. Consider a stack (which is a last-in, first-out list). You might have a program that requires three types of stacks. One stack is used for integer values, one for floating-point values, and one for characters. The algorithm that implements each stack is the same, even though the data being stored differs. In a non-object-oriented language, you would be required to create three different sets of stack routines, with each set using different names. However, because of polymorphism, in Java you can specify a general set of stack routines that all share the same names. More generally, the concept of polymorphism is often expressed by the phrase “one interface, multiple methods”. This means that it is possible to design a generic interface to a group of related activities. This helps reduce complexity by allowing the same interface to be used to specify a general class of action. It is the compiler’s job to select the specific action (that is, method) as it applies to each situation.
## Unique Word Frequency

<table>
<thead>
<tr>
<th>Word</th>
<th>count</th>
<th>Word</th>
<th>Count</th>
<th>Word</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>stack</td>
<td>5</td>
<td>differs</td>
<td>1</td>
<td>Polymorphism</td>
<td>1</td>
</tr>
<tr>
<td>interface</td>
<td>4</td>
<td>determine</td>
<td>1</td>
<td>point</td>
<td>1</td>
</tr>
<tr>
<td>used</td>
<td>3</td>
<td>design</td>
<td>1</td>
<td>phrase</td>
<td>1</td>
</tr>
<tr>
<td>same</td>
<td>3</td>
<td>data</td>
<td>1</td>
<td>oriented</td>
<td>1</td>
</tr>
<tr>
<td>general</td>
<td>3</td>
<td>create</td>
<td>1</td>
<td>object</td>
<td>1</td>
</tr>
<tr>
<td>each</td>
<td>3</td>
<td>Consider</td>
<td>1</td>
<td>non</td>
<td>1</td>
</tr>
<tr>
<td>action</td>
<td>3</td>
<td>concept</td>
<td>1</td>
<td>nature</td>
<td>1</td>
</tr>
<tr>
<td>--------------</td>
<td>-------</td>
<td>----------</td>
<td>-------</td>
<td>---------------</td>
<td>-------</td>
</tr>
<tr>
<td>--------------</td>
<td>-------</td>
<td>----------</td>
<td>-------</td>
<td>---------------</td>
<td>-------</td>
</tr>
<tr>
<td>--------------</td>
<td>-------</td>
<td>----------</td>
<td>-------</td>
<td>---------------</td>
<td>-------</td>
</tr>
<tr>
<td>--------------</td>
<td>-------</td>
<td>----------</td>
<td>-------</td>
<td>---------------</td>
<td>-------</td>
</tr>
<tr>
<td>--------------</td>
<td>-------</td>
<td>----------</td>
<td>-------</td>
<td>---------------</td>
<td>-------</td>
</tr>
<tr>
<td>--------------</td>
<td>-------</td>
<td>----------</td>
<td>-------</td>
<td>---------------</td>
<td>-------</td>
</tr>
<tr>
<td>--------------</td>
<td>-------</td>
<td>----------</td>
<td>-------</td>
<td>---------------</td>
<td>-------</td>
</tr>
<tr>
<td>values</td>
<td>2</td>
<td>compiler</td>
<td>1</td>
<td>methods</td>
<td>1</td>
</tr>
<tr>
<td>specify</td>
<td>2</td>
<td>characters</td>
<td>1</td>
<td>method</td>
<td>1</td>
</tr>
<tr>
<td>specific</td>
<td>2</td>
<td>applies</td>
<td>1</td>
<td>means</td>
<td>1</td>
</tr>
<tr>
<td>situation</td>
<td>2</td>
<td>allows</td>
<td>1</td>
<td>meaning</td>
<td>1</td>
</tr>
<tr>
<td>set</td>
<td>2</td>
<td>allowing</td>
<td>1</td>
<td>list</td>
<td>1</td>
</tr>
<tr>
<td>routines</td>
<td>2</td>
<td>algorithm</td>
<td>1</td>
<td>last</td>
<td>1</td>
</tr>
<tr>
<td>polymorphis</td>
<td>2</td>
<td>language</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>names</td>
<td>2</td>
<td>activities</td>
<td>1</td>
<td>job</td>
<td>1</td>
</tr>
<tr>
<td>different</td>
<td>2</td>
<td>actions</td>
<td>1</td>
<td>Java</td>
<td>1</td>
</tr>
</tbody>
</table>
3.3 Experimentation

The results at the end of this section supports the central idea of this entire process that the savings in terms of time by reading summaries can be invaluable, particularly in the case of Digital Libraries, Research Paper / Text Book Summarization.

Here, the authors’ conception was applied to a corpus of 50 texts which varied in length from between three and 2500 Sentences. These texts
were randomly chosen from Digital Libraries & on-line archives. They include both scientific and general press articles, extracts from books, and internal research notes. For the purpose of experimentation, one of the huge books, ‘The JavaTM 2 Complete Reference – 5th edition by Herbert Schildt’ is taken from ‘www.osborne.com’. Couple of them is taken from ‘www.free-ebooks.net’ and some of them from ‘www.planetpdf.com’ and related web sites. The coding part of these books, if any, is removed before giving it to the process of summary generation. Validation was carried out by two jurors, who were not specialists in the fields concerned, who read all the 50 source texts and prepared their own summaries not changing the original sentences. The source texts were given to the Digital Library of India’s Text Summarization Utility [97] to get a summary for comparison. And then, the summaries generated by the 'automated method' were compared with those prepared by these two jurors and also with those generated by the above method. Then results are estimated in terms of 'percentage of accuracy'. The average size of the summary is approximately 15-20% of the source text.

The method that is discussed here does some mechanical iteration over the entire text and then maps the associability of frequently repeating words with the length of the sentence and the number of such words in each sentence. Obviously, if the writer of the text is discussing some technical feature, it is very evident in the results that the most frequently repeating words definitely helps in preparing a good
summary. Incidentally, this method is highly effective in summarizing 'technical' content and the effectiveness in summarizing 'general' stuff is also satisfactory.

Assuming the summaries (prepared by our jurors/prepared by DLI’s Text Summ.Util.) as “noted”: we will define a summary as being "very efficient" if more than 80% of the sentences which make it up are among the "noted" ; less than 10% are irrelevant ; and less than 20% from the "noted" are missing : we will define a summary as being "efficient" if between 60-80% of the sentences which make it up are among the "noted" ; less than 20% of it are irrelevant ; and 20-30% from the "noted" are missing : we will define a summary as being "moderately efficient" if between 40% and 60% of the sentences which make it up are among the "noted"; less than 30% are irrelevant ; and 30-40% from the "noted" are missing we will define a summary as being "inefficient" in all other cases.

They are carefully selected so as to cover different subjects like 'technology', 'news', 'statistical', 'medical' etc. The results are summarized and categorized on the length of the documents and the 'efficiency grades' are given as decided above.

Test Results with respect to the summaries prepared by jurors:
<table>
<thead>
<tr>
<th>File-Length (Range in sentences)</th>
<th>no. of files tested</th>
<th>Average of the Accuracy %</th>
<th>Grades endorsed</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 100</td>
<td>38</td>
<td>52</td>
<td>moderately efficient</td>
</tr>
<tr>
<td>400-500</td>
<td>10</td>
<td>57</td>
<td>moderately efficient</td>
</tr>
<tr>
<td>2300-2500</td>
<td>2</td>
<td>71</td>
<td>efficient</td>
</tr>
</tbody>
</table>

Test Results with respect to the summaries prepared by MEAD method using the Text Summarization Utility provided by the DLI:

<table>
<thead>
<tr>
<th>File-Length (range in sentences)</th>
<th>no. of files tested</th>
<th>Average of the Accuracy %</th>
<th>Grades endorsed</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 100</td>
<td>38</td>
<td>72</td>
<td>efficient</td>
</tr>
<tr>
<td>400-500</td>
<td>10</td>
<td>67</td>
<td>efficient</td>
</tr>
<tr>
<td>2300-2500</td>
<td>2</td>
<td>73</td>
<td>efficient</td>
</tr>
</tbody>
</table>

3.4 Discussion

For the purpose of comparison of our results, we have matched our results with those generated by the 'Digital Library of India's Text Summarization Utility, which is in turn powered by 'MEAD', a universal standard in automatic evaluation as a unit of measurement.

It is observed that the Text Summarization Utility of the Digital Library of India failed in extracting the following points:

1. It removes non-alphanumeric characters like '+' from the source text. This made it particularly prone to giving "wrong summaries". Ex: When we have tested on an extract of 'C++
Standard Template Libraries’, it summarized the text as 'C Standard Template Libraries (STL)'. This is not at all acceptable as the language C doesn't have STL's.

2. It removes characters '(' and ')' and so when technical stuff like the 'malloc ()' functions of 'C-language' are discussed; it says only 'malloc' which is not advised.

3. It is always giving the first paragraph of the source text into the summary, which is sometimes irrelevant.

4. It reads the 'new line' character i.e., '\n' as a terminator and so when the source text doesn't have '\n's in between sentences, all of them are evaluated into the summary, without checking for relevancy. That is if your text document contains one single paragraph which is very large, the entire document is given as the summary.

Where we have improved:

Rectifying all the four points discussed above, we have improvised our method, to give the best possible summaries. Moreover, we are giving the user to opt for the cut-off marks which gives him the facility to alter his summaries according to his wish. If he changes the cut-off marks and runs the procedure, he will be given an entirely different summary.
3.5 Conclusion

While the summaries prepared by jurors have differed slightly with those generated by us and also by the 'MEAD' method, it is to be noted that our summaries are very 'similar' to those generated by the 'MEAD' method. In fact, in some areas as explained above, our method is observed to be generating better 'summarizations' than that of 'MEAD'.

By seeing the above results it can be concluded as follows:

1. The above method works at its best in summarizing large text documents that are having more than 2000 sentences.

2. The above method works moderately efficient in summarizing text documents that are having less than 1000 sentences.

3.6 Limitations

Even though the results are observed to be positive and satisfactory in our method, the entire process we have developed suffers from the following limitations:

1. Time complexities are not considered while comparing with the 'MEAD' method.

2. We could not spare enough time & effort to test more number of documents.

3. 'Code efficiency' of our algorithm is not checked and verified.