**Text Summarizer**

*Synopsis for Final Year Project*

***Department of Computer Science***



| **Submitted To**  HOD Dr. Sonia Juneja  Mentor Ms. Dhanshri Parihar  CS Department  IMSEC, Ghaziabad | **Submitted By:**  Abhimanyu Sharma(1901430120002)  Ashutosh(1901430120013)  Avinash Chaurasiya(1901430120015) Kuldeep Saini(1901430120028) |
| --- | --- |



**Project Title:** Text Summarizer



## Project Description

Text Summarizer will be a web application where you can get extract-based summaries of drastically lengthy text. Text can be extracted from images or directly typed. It also allows the conversion of resumes into different languages.

This project will be developed using react js for the front end and Django framework for the back end. The server will be implemented in the rest architecture.

Input will be taken from the user interface and sent to the server as text or images. In the case of an embodiment, the server will extract the text using OCR technologies which will be implemented using third-party API. Then input text will be processed in our algorithm. This algorithm provides extract-based summaries. In the last step, the desired result will be sent to the client and shown to the user.

## Literature Survey

With the dramatic growth of the Internet, people are overwhelmed by the tremendous amount of online information and documents. This expanding availability of documents has demanded exhaustive research in the area of automatic text summarization. According to Radef et al. [6] a summary is 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”. Automatic text summarization is the task of producing a concise and fluent summary while preserving key information content and overall meaning. In recent years, numerous approaches have been developed for automatic text summarization and applied widely in various domains. For example, search engines generate snippets as previews of the documents. Other examples include news websites that produce condensed descriptions of news topics usually as headlines to facilitate browsing or knowledge extractive approaches. Automatic text summarization is very challenging, because when we as humans summarize a piece of text, we usually read it entirely to develop our understanding, and then write a summary highlighting its main points. Since computers lack human knowledge and language capability, it makes automatic text summarization a very difficult and non-trivial task. Automatic text summarization gained attraction as early as the 1950s. An important research of these days was [7] for summarizing scientific documents. Luhn et al. [7] introduced a method to extract salient sentences from the text using features such as word and phrase frequency. They proposed to weight the sentences of a document as a function of high-frequency words, ignoring very high-frequency common words. Edmundson et al. [8] described a paradigm based on key phrases which in addition to standard frequency-depending weights, used the following three methods to determine the sentence weight:

(1) Cue Method: The relevance of a sentence is calculated based on the presence or absence of certain cue words in the cue dictionary

(2) Title Method: The weight of a sentence is computed as the sum of all the content words appearing in the title and headings of a text.

(3) Location Method: This method assumes that sentences appearing at the beginning of a document as well as the beginning of individual paragraphs have a higher probability of being relevant.

Since then, many works have been published to address the problem of automatic text summarization (see [9, 10] for more information about more advanced techniques until the 2000s). In general, there are two different approaches for automatic summarization: extraction and abstraction. Extractive summarization methods work by identifying important sections of the text and generating them verbatim; thus, they depend only on the extraction of sentences from the original text. In contrast, abstractive summarization methods aim at producing important material in a new way. In other words, they interpret and examine the text using advanced natural language techniques in order to generate a new shorter text that conveys the most critical information from the original text. Even though summaries created by humans are usually not extractive, most of the summarization research today has focused on extractive summarization. Purely extractive summaries oftentimes give better results compared to automatic abstractive summaries [2]. This is because of the fact that abstractive summarization methods cope with problems such as semantic representation, inference, and natural language generation which are relatively harder than data-driven approaches such as sentence extraction. As a matter of fact, there is no completely abstractive summarization system today. Existing abstractive summarizers often rely on an extractive preprocessing component to produce the abstract of the text [11, 33]. Consequently, we focus on extractive summarization methods and discuss an overview of some of the most dominant approaches in this category.

As mentioned before, extractive summarization techniques produce summaries by choosing a subset of the sentences in the original text. These summaries contain the most important sentences of the input. Input can be a single document or multiple documents. In order to better understand how summarization systems work, we describe three fairly independent tasks that all summarizers perform [1]:

1) Construct an intermediate representation of the input text which expresses the main aspects of the text.

2) Score the sentences based on the representation.

3) Select a summary comprising a number of sentences

Every summarization system creates some intermediate representation of the text it intends to summarize and finds salient content based on this representation. There are two types of approaches based on representation: topic representation and indicator representation. Topic representation approaches to transform the text into an intermediate representation and interpret the topic(s) discussed in the text. Topic representation-based summarization techniques differ in terms of their complexity and representation model, and are divided into frequency-driven approaches, topic word approaches latent semantic analysis, and Bayesian topic models [1]. Indicator representation approaches describe every sentence as a list of features (indicators) of importance such as sentence length, position in the document, having certain phrases, etc.

When the intermediate representation is generated, we assign an importance score to each sentence. In topic representation approaches, the score of a sentence represents how well the sentence explains some of the most important topics of the text. In most of the indicator representation methods, the score is computed by aggregating the evidence from different indicators. Machine learning techniques are often used to find indicator weights.

Eventually, the summarizer system selects the top k most important sentences to produce a summary. Some approaches use greedy algorithms to select the important sentences and some approaches may convert the selection of sentences into an optimization problem where a collection of sentences is chosen, considering the constraint that it should maximize overall importance and coherency and minimize the redundancy. There are other factors that should be taken into consideration while selecting important sentences. For example, the context in which the summary is created may be helpful in deciding its importance. The type of the document (e.g. news article, email, scientific paper) is another factor that may impact selecting the sentences.

When assigning weights of words in topic representations, we can think of binary (0 or 1) or real-value (continuous) weights and decide which words are more correlated to the topic. The two most common techniques in this category are word probability and TFIDF (Term Frequency Inverse Document Frequency). The simplest method to use the frequency of words as an indicator of importance is word probability. The probability of a word w is determined as the number of occurrences of the word, f (w), divided by the number of all words in the input (which can be a single document or multiple documents).

Since word probability techniques depend on a stop word list in order to not consider them in the summary and because deciding which words to put in the stop list is not very straightforward, there is a need for more advanced techniques. One of the more advanced and very typical methods to give weight to words is TFIDF (Term Frequency Inverse Document Frequency). This weighting technique assesses the importance of words and identifies very common words (that should be omitted from consideration) in the document(s) by giving low weights to words appearing in most documents.

**Problem Definition**

Traditionally, summarization has been mostly applied to two genres of text: scientific papers and news stories. These genres are distinguished by a high level of stereotypical structure. In both these domains, simply choosing the first few sentences of a text or texts provides a baseline that few systems can better and none can better by much. Attempts to summarize other texts, e.g., fiction or email, have been somewhat less successful.

Recently, summarization researchers have also investigated methods of text simplification (or compression). Typically, these methods apply to a single sentence at a time. Simple methods include dropping unimportant words (determiners, adverbs). Complex methods involve reorganizing the syntactic parse tree of the sentence to remove sections or to rephrase units in shorter forms. Language modeling approaches in TS have mostly focused on this method.

The increasing availability of online information has triggered intensive research in the area of automatic text summarization within Natural Language Processing (NLP). Text summarization reduces the text by removing the less useful information which helps the reader to find the required information quickly. There are many kinds of algorithms that can be used to summarize the text. One of them is TF-IDF (Term Frequency-Inverse Document Frequency). It generates the summary with 67% of accuracy, which is a better result of the summary than other online summarizers[2].

**Ghantt chart**

| **Task Name** | **Sep 22** | **Oct 22** | **Nov 22** | **Dec 22** | **Jan 23** | **Feb 23** | **Mar 23** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Planning** | **20 Sep** |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| **Research** | **25 Sep** |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| **Design** |  |  | **25 Nov** |  |  |  |  |
|  |  |  |  |  |  |  |  |
| **Implementation** |  |  |  | **30 De**c |  |  |  |
|  |  |  |  |  |  |  |  |
| **Testing** |  |  |  |  | **30 Jan** |  |  |
|  |  |  |  |  |  |  |  |
| **Deployment** |  |  |  |  |  | **28 Feb** |  |

**Implementation Methods**

TF-IDF is a numerical statistic which reflects on how important a word is to a document in the collection or corpus (Saltonet al., 1988). The TF-IDF value increases proportionally to the number of times when a word appears in the document, but it is offset by the frequency of the word in the corpus, which helps to control the fact that some words are more common than others. The frequency term means the raw frequency of a term in a document. Moreover, the term regarding inverse document frequency is a measure of whether the term is common or rare across all documents in which can be obtained by dividing the total number of documents by the number of documents containing the term [3].

This program is divided into three main functions which are preprocessing, feature extraction, and summarization. After the document is inputted into the program, the preprocessing function splits the text in to a list of words using tokenization functions. These tokenization functions are divided into two which are sentence tokenization and word tokenization. Sentence tokenization is a function to split the paragraph into sentences. While word tokenization is a function to split the string of written language into words and punctuation.

After that, the sentences are tokenized into a list of words. To make sure no unnecessary word in the list, every word in the list are classified using POS tagger function. Only VERB and NOUN are calculated in this experiment, because these types of the word are biased to make a summary [4] . All stopwords and clitics are also removed to prevent ambiguities. Then, the list of words is processed using stemming function to normalize the words by removing affixes to make sure that the result is the known word in the dictionary [5].

From the preprocessed list of words, the TF-IDF value of each noun and verb can then be calculated. The equation of TF-IDF can be seen below.



The value of TF-IDF ranges from zeroto one with ten-digit precision. After been calculated,

these words are sorted in descending order by its value. Then, it is compiled into the new dictionary of word and its value. This sorting is important to analyze the rank of TF-IDF value from all of the words to check the output summary. After knowing TF-IDF value of each word, it can calculate the importance value of a sentence. The importance value of a sentence is a sum of the value of every noun and verb in the sentence. Every sentence in the document is sorted in descending order. Finally, three to five sentences with the highest TF-IDF value are chosen. The number of sentences in the final summary may change depending on the compression rate of the program chosen by the user.

**Requirements**

**Server Side:**

* Operating System: Windows.
* Processor: 1 CPU core.
* 1GB RAM.

**Client Side:**

* A reliable internet connection. ADSL / Broadband connections are recommended.
* Operating System: Any OS supporting web browser.
* Web Browser: Mozilla Firefox, Google Chrome, Safari, etc
* Processor: Dual Core 1.6 GHz.
* 256MB RAM.
* Microsoft Office 2007.



## References

[1]. Ani Nenkova and Kathleen McKeown. 2012. A survey of text summarization techniques. In Mining Text Data. Springer, 43–76.

[2]. Hans Christian, Mikhael Pramodana Agus and Derwin Suhartono, “Single document automatic text summarization using term frequency-inverse document frequency (tf-idf)”:

<https://journal.binus.ac.id/index.php/comtech/article/view/3746/3037>

[3]. Munot, N., & Govilkar, S. S. (2014). Comparative study of text summarization methods. International Journal of Computer Applications,102(12), 33-37.

[4]. Yohei, S. (2002). Sentence extraction by TF/IDF and Position Weighting from newspaper articles. In Proceedings of the Third NTCIR Workshop.

[5]. Bird, S., Klein, E., & Loper, E. (2009) Natural language processing with Python. United States: O'Reilly Media.

[6]. Dragomir R Radev, Eduard Hovy, and Kathleen McKeown. 2002. Introduction to the special issue on summarization. Computational linguistics 28, 4 (2002), 399–408.

[7]. Hans Peter Luhn. 1958. The automatic creation of literature abstracts. IBM Journal of research and development 2, 2 (1958), 159–165.

[8]. Harold P Edmundson. 1969. New methods in automatic extracting. Journal of the ACM (JACM) 16, 2 (1969), 264–285.

[9] Günes Erkan and Dragomir R Radev. 2004. LexRank: Graph-based lexical centrality as salience in text summarization. J. Artif. Intell. Res.(JAIR) 22, 1 (2004), 457–479.

[10] Yihong Gong and Xin Liu. 2001. Generic text summarization using relevance measure and latent semantic analysis. In Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 19–25

## Contact details of Team Members

| **S.**  **No.** | **Name** | **Contact No** | **Email ID** | **Father’s contact detail** |
| --- | --- | --- | --- | --- |
| 1 | Kuldeep Saini | 9119037006 | mailtokuldeepsaini@gmail.com | 8448489881 |
| 2 | Abhimanyu Sharma | 8512899097 | abhimanyusharma2k@gmail.com | 8510042225 |
| 3 | Ashutosh | 9315988320 | ashutosh.sinha1109@gmail.com | 7011482939 |
| 4 | Avinash Chaurasiya | 9918707047 | achaurasiya1008@gmail.com |  |

**Project Guide Detail:**

| **S. No.** | **Guide Name** | **Email Id** | **Contact Number** |
| --- | --- | --- | --- |
| 1 | Miss Dhanshri  Parihar | dhanshri.parihar@imsec.ac.in | 01204940000 |

**Signature of Project Guide with date:**

# SPECIFICATIONS FOR SYNOPSIS

1. The synopsis shall be computer typed (English- British, Font -Times Roman, Size-12 point) and printed on A4 size paper.
2. The Synopsis shall be typed on one side only with 1.5 spacing with a margin of 2.5 cm on the left, 2.5 cm on the top, and 1.25 cm on the right and at the bottom.
3. The diagrams should be printed on a light/white background, Tabular matter should be clearly arranged. A decimal point may be indicated by a full stop(.)The caption for the Figure must be given at the BOTTOM of the Fig. and the Caption for the Table must be given at the TOP of the Table.