Proceedings of the Fourth International Conference on Trends in Electronics and Informatics (ICOEI 2020)
IEEE Xplore Part Number: CFP20J32-ART; ISBN: 978-1-7281-5518-0

Extractive Text Summarization from Web pages using Selenium and TF-IDF algorithm

K Usha Manjari, Syed Rousha, Dasi Sumanth, Dr. J Sirisha Devi
Department of Computer Science and Engineering
Institute of Aeronautical Engineering, JNTU(H)
Hyderabad, India

Abstract: **To obtain an overview of the content present in numerous documents, is a time-consuming task. Similarly, searching for specific information online, from multiple websites and webpages is a monotonous task. To avoid this, automatic text summarization is one of the most widely adopted techniques today to get a concise and brief outline of the information. In this paper, a novel process is proposed to generate an extractive summary of the information based on the user's query by extracting data from multiple websites over the internet. Web-scraping through Selenium is also discussed. The Term Frequency–Inverse Document Frequency (TF-IDF) algorithm is applied for text summarization. The proposed approach is unique and efficient for generating summaries as per the user's request.**

Keywords: summarization, web pages, selenium, Term Frequency–Inverse Document Frequency (TF-IDF) algorithm, extractive.

I. INTRODUCTION

With the gradual advancements in technology, more and more computers are being used, which results in the production of enormous volumes of data and information. The internet now has a glut of information on any aspect that one often feels so overwhelmed just by the magnitude of it. There are massive repositories of data on the internet, which are just a search away. A web user attempting to perceive information on a particular aspect is most likely to visit a popular search engine and browse through the web pages listed in the results. This will cost the user a lot of time. Visiting back and forth from one page to another and looking for pieces of information is a mundane task. The webpages obtained from the search results have huge chunks of data giving rise to information overload. Therefore, there is a requirement of a system that could retrieve the data from various accessible sources from the web, aggregate and summarize the information gathered into a concise summary and present it to the user. Using text summarization on the vast amount of data available on different web pages on the internet can make this possible. Text summarization is a technique that is used to summarize large voluminous texts into a summary concentrating on the sectors that convey important information, without losing the key insights present in the data. Automatic text summarization intends to transfigure verbose records into diminished variants; this is something that could be stressful and highly inefficient if done manually. Computer learning algorithms can be instructed to embrace documents and classify the segments that carry vital details and data before presenting the expected summarized text. Depending upon the way of summarizing, text summarization has two different approaches to it, i.e., extractive text summarization and abstractive text summarization. In the extraction based text summarization, the text from the document is directly extracted, and summaries are made from the extracted data. In contrast, in the abstractive text summarization, the content in the original document is paraphrased such that it still holds the abstract of the original document but in a shorter form. In this paper, the extractive text summarization is focussed more as use it in the system being proposed. In extractive text summarization, the content is neither changed nor paraphrased. Few sentences which can be used to index the document, some important key-phrases, etc. are extracted in this technique. Another critical method that is used in the culmination of the proposed system is web scraping. Web scraping is the technique that is used to extract the data from the web pages. It can be done through various means, numerous libraries, and frameworks that contain several functions that facilitate web scraping. Selenium is an open-source web-based automation tool which is quite efficient for web scraping. The web driver in Selenium provides numerous features that enable us to navigate through the desired web pages and fetch various contents of the page depending upon our necessities. Thus lots of data from various web pages concerning the user's query can be extracted from multiple web pages and grouped. This consolidated data can be effectively analysed and summarized such that the final summary contains all the necessary details and critical features of the data, in proportion with the specified word limit. Here the objective of our proposed work is to generate an extractive summary of the information based on the user's query by extracting data from multiple websites over the internet, which will save the user much time and effort to go on to different web pages repeatedly and look out for relevant information.

The remaining paper is arranged in the following way- Previous works related to web scraping and text

summarization are discussed in section 2. Section 3 describes the complete methodology of the proposed system. Section 4 consists of the results obtained. Conclusion and future work are stated in the last part.

II. RELATED WORK

Text Summarization is gaining popularity from quite a few years. Earlier work-related to text summarization, web scraping, different approaches used for summarization are as follows: In [1], an overview of web scraping is given, and various web scraping techniques are discussed and compared. It also gives a note of some web scraping software being used. An outline of text summarization with regards to information interpretation and retrieval is presented in [2]. It discusses the web characteristics, text summarization, summarizing the web pages, and usage of hyperlinks for summarizing web pages. Focuses on automation of the website summarization is explained in [3]. Machine learning and natural language processing techniques are applied. The paper compares and summarizes automatically generated, to the manually generated summaries. From [4], the usage and application of the Selenium web driver is inferred. An overview of the implementation of Selenium IDE with a web driver is presented in [5]. The paper explains the integration of Selenium IDE with a web driver since only Selenium is not compatible with all web browsers. Web pages summarization using Latent Semantic Analysis is given in [6]. The paper clearly explains Latent Semantic Analysis, representing web pages, dealing with text fragments of the web pages, and an overall idea about text summarization of web pages based on the user's request. A list of various approaches for text summarization, which include cluster-based, graph-based, and latent semantic analysis based, term frequency-based, etc. is presented in [7]. The paper also gives an idea about multi-document text summarization. It compares the approaches mentioned above for text summarization giving us a good insight into the efficiency of each technique. An overview of summarizing multiple documents using the TF-IDF algorithm is presented in [8]. It applies the notion of ontology, which deals with the relationship between multiple entities. Explanation about performing text summarization on a single document using the Term Frequency- Inverse Document Frequency (TF-IDF) algorithm is presented in [9]. The paper also compares the proposed approach with other online summarizers and shows the accuracy of the proposed approach. The application or usage of the TF-IDF algorithm for finding the word relevance in various document queries is given in [10]. It is used to determine which words are more important in a given set of documents. The paper points out the problems in query retrieval, an overview of the TF-IDF algorithm, and compares the experimental results. The implementation of text summarization with the help of NLTK (Natural Language Tool Kit) using the TF-IDF algorithm is presented in [11]. Automation testing implemented using Selenium WebDriver tool is described in [12]. The entire procedure for implementation is explained stepwise.

III. METHODOLOGY

A. User Query

For searching the information related to any topic over the internet, a user query has to be given. A query substantially reduces the effort to browse content and gives us the results most appropriate to the user requirement. It can be a word or a group of words related to the information that needs to be searched. These queries are entered in plain text format by the user.

B. Data extraction through Selenium

Web scraping is an efficient technique that is used for extracting huge amounts of data from various websites. This data can then be stored in any file or a database. Web scraping involves the extraction of the HTML code of the web page. Here Selenium is used for fulfilling the purpose. Generally, Selenium is used for web browser automation and is an automated testing tool. Nevertheless, it can as well be used for web scraping.

C. Procedure for extraction

The flow chart for the steps involved in the data extraction process is given below:

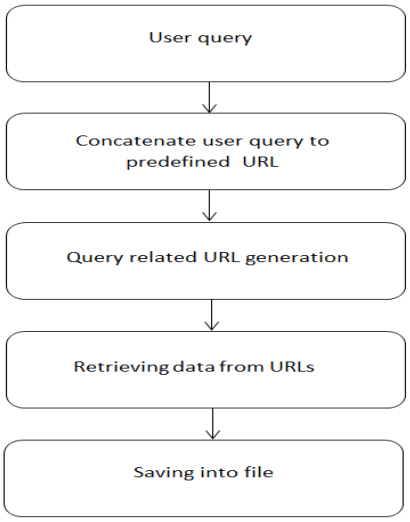


Fig 1: Data extraction through Selenium.

Initially, the user enters the query for performing the search operation.

Query related URL generation:

The user's query is taken and depending upon the query, the URLs of the most popular web pages associated with the information related to the query are scraped, this can be done with the help of an external search engine like Google, Bing, etc. wherein the user's query is concatenated to a predefined URL concerned with the search engine, and then the URLs of the topmost web pages are acquired as such the results of any query on a search engine.

Retrieving data from URLs:

The obtained URL is then used for carrying out the search operation using Selenium. With the help of the Selenium web driver, the websites' URLs related to the search query are obtained. Thereafter, each URL is opened automatically and the data in each webpage is scraped one after the other.

Saving into the file:

All of this data is then stored in a user-specified file locally on a computer or in the form of tables in a database. Here the data is stored in a text file. Text summarization is performed over this file.

D. Summarization using TF-IDF Algorithm

The previous steps involve scraping and gathering the data from various websites, which results in the formation of a massive repository of information. Here there are high chances that some of the information may be repetitive. Therefore, going through all of the data gathered will be tough and a monotonous task. There is a need to adequately filter and skim out the essential and insignificant aspects, respectively, from the gathered data. The most efficient technique for achieving this is Text summarization. Text summarization is done on significantly voluminous texts to convert them into small and straightforward summaries. There are two different approaches for performing text summarization, Extractive text summarization, and Abstractive text summarization. In extractive text summarization, the text from the original input draft, which is to be summarized, is directly extracted into the summary. Whereas in abstractive text summarization, the text from the original draft is slightly modified and paraphrased as per the requirements. Here the TF-IDF algorithm for performing extractive text summarization is used on the gathered data. The TF-IDF algorithm makes use of two different parameters, namely, Term frequency and Inverse document frequency. These two parameters are defined upon the content of the document adequately, which facilitates the classification of the content and then summarizes accordingly.

The TF-IDF algorithm can be implemented in any programming language. Here as the summarization is being done in association with Web scraping, this combined framework can be implemented in Python with great ease due to its support for availability and integration of a variety of libraries. Libraries like selenium, NLTK can be used for amplifying the efficiency of the Web scraping and text summarization, respectively.

Steps involved in TF-IDF algorithm

1. ***Tokenize the sentences***

Tokenization is dividing large blocks of texts into smaller units, generally called as Tokens. For instance, a word is a token in a sentence. Similarly, a sentence is a token in a paragraph. Initially, the units of the content, i.e., sentences are tokenized, and weights are assigned to these sentences. The Weight assigned to a token can depend on various parameters such as frequency, uniqueness, etc. To tokenize, use the sub-module of the tokenize module present in NLTK is used. At the end of this step, a matrix containing all the identified tokens with their corresponding weights is obtained.

2. Calculation of Term Frequency (TF)
After obtaining the weight matrix, the Term Frequency (TF) for each word in a paragraph is calculated.
Definition of TF,
TF(term) = (Number of times term appears in a document) / (Total number of terms in the document)
The TF values of each word are stored in a matrix which is called the TF matrix.

3. Calculation of Inverse Document Frequency (IDF):
The Inverse Document Frequency (IDF) for each word in a paragraph is calculated. Here, the document is assumed as a paragraph and a term as a word in the corresponding paragraph.
Definition of IDF,
IDF(term) = log_e(Total number of documents / Number of documents with term in it).
The IDF values of each word are stored in a matrix which is called the IDF matrix.

4. Calculation of TF-IDF Score
With both the matrices of the individual TF and IDF scores, the combined TF-IDF score of each word is calculated. This is obtained by multiplying the word's TF score with it's IDF score.

TF-IDF(term) = TF(term) * IDF(term)

5. Scoring the sentences

After calculating the TF-IDF score of each word, now the sentences containing these words are to be scored. Scoring a

sentence is different in various algorithms. Here, the use of the TF-IDF score of the words in a sentence to calculate the sentence score is made. The average of TF-IDF value of all the words belonging to a particular sentence is calculated, which becomes the score of that sentence.

Sentence score = (Sum of TF-IDF values of all the words in a sentence)/Total number of words in the sentence.

6. Finding the threshold

After calculating the TF-IDF score of each word, the sentences containing these words are to be scored. Scoring a sentence is different in various algorithms. Here, the use of the TF-IDF score of the words in a sentence to calculate the sentence score is made. The average of TF-IDF value of all the words belonging to a particular sentence is calculated, which becomes the score of that sentence.

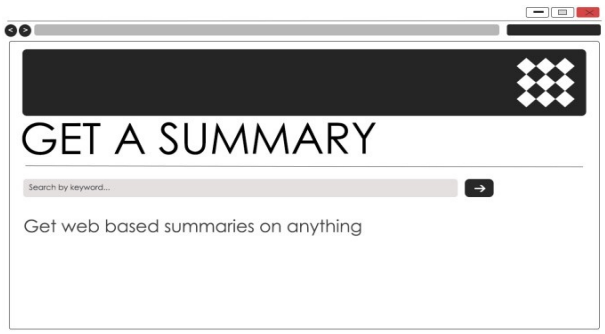
7. Generating the summary

The summary is generated depending upon the threshold score. All the sentences possessing a TF-IDF score greater than the threshold score are admitted into the summary, and the rest are ignored. Thus the final summary of the gathered data is obtained.

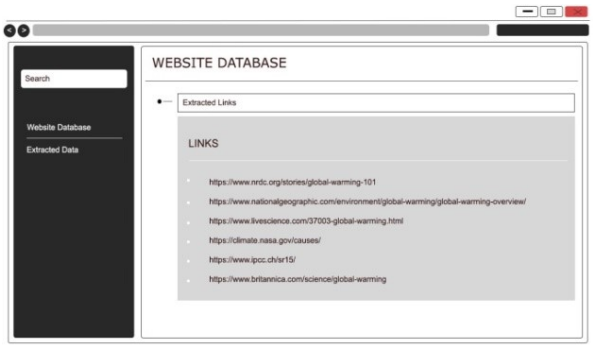
IV. RESULTS

The following is obtained as a result of the text summarization process:

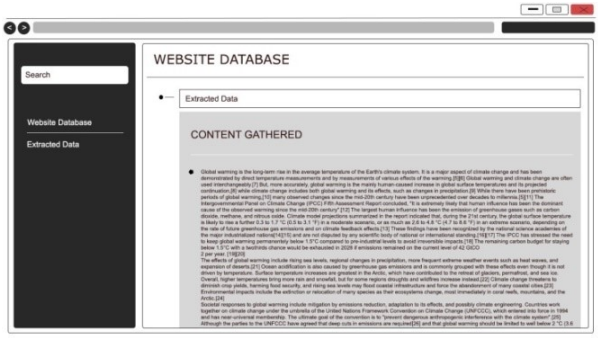
Step 1:
User interface:
The web page which acts as an interface for the user to enter the query.



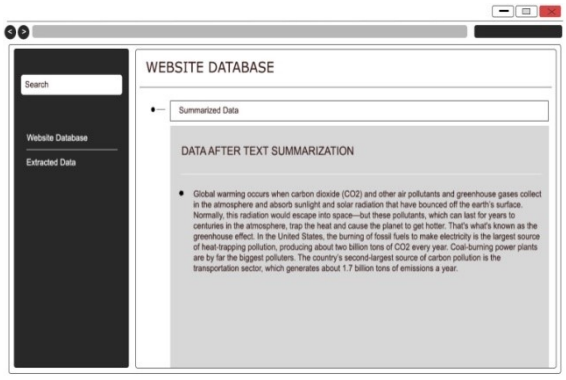
Step 2:
Extracted URLs:
The query fetched from the search box is then used for obtaining the related web results. The related web URLs are extracted.



Step 3:
Extracted data:
Each URL is opened using selenium and the content present in each of them is scraped.



Step 4:
Final output:
Summarization is performed on the data extracted in the previous step. The final summary is thus obtained.



V. CONCLUSION AND FUTURE WORK

Summarizing the content of multiple web pages related to a particular user query helps in obtaining a quick gist of the topic searched. To retrieve the data, web scraping using Selenium is performed. However, other web scraping techniques can also be applied. Here the TF-IDF algorithm is

used for summarizing the data. Other text summarization algorithms, such as graph-based, centroid based, and knowledge-based, etc., can as well be applied. The approach used here is extractive summarization; future work can be done on performing a similar process through abstractive summarization.

VI. REFERENCES

[1] SCM de Sirisuriya, "A Comparative Study on Web Scraping," Proceedings of 8th International Research Conference, KDU, Published November 2015.

[2] Alvaro Mendes Barbosa, "Overview of text summarization in the context of information retrieval and interpretation: Applications for web pages summarization."

[3] Y. Zhang, N. Zincir-Heywood, and E. Milios, "Summarizing Web Sites Automatically."

[4] Sherry Singla, Harpreet Kaur, "Selenium Keyword Driven Automation Testing Framework", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 4, Issue 6, June 2014.

[5] Nidhika Uppal, Vinay Chopra, "Design and Implementation in Selenium IDE with Web Driver", International Journal of Computer Applications, Volume 46-No.12 May 2012.

[6] J. Guadalupe Ramos, Isela Navarro-A Latorre, Georgina Flores Becerra, Omer Flores-Sanchez, "A Formal Technique for Text Summarization from Web Pages by using Latent Semantic Analysis", ISSN 1870-4069.

[7] Chintan Shah, Anjali G. Jivani, "Literature Study on Multi-document Text Summarization Techniques", International Conference on Smart Trends for Information Technology and Computer Communications, pp 442-451.

[8] Prof. Amit Savyanavar, Bhakti Mehta, Varsha Marathe, Priyanka Padvi and Manjusha Shewale, "Multi-Document Summarization Using TF-IDF Algorithm", International Journal of Engineering And Computer Science ISSN: 2319-7242, Volume 5, Issue 4, April 2016, Page No. 16253-16256.

[9] Hans Christian, Mikhael Pramodana Agus, Derwin Suhartono, "Single Document Automatic Text Summarization Using Term Frequency-Inverse Document Frequency (TF-IDF)", ComTech Vol.7, No 4, December 2016: 285-294.

[10] Juan Ramos, "Using TF-IDF for Word Relevance in Document Queries."

[11] NV Ganapathi Raju Y. Sri Lalitha, J Sirisha Devi, L. Sukanya, "Analysis of Parts of Speech Tagging on Text Clustering", International Journal of Innovative Technology and Exploring Engineering (IJITEE), Vol.8, No 8, June 2019: 2287-2291.

[12] Satish Gojare, Rahul Joshi, Dhanashree Gaigaware, “Analysis and Design of Selenium WebDriver Automation Testing Framework”, ScienceDirect, Procedia Computer Science 50(2015) 341 – 346.