***Summary Generator Using NLP & PYQT***

Pratyush Garg  
*Dept. Computer Science and Engineering   
UGC Affiliated*  
*HMR Institute of Technology and Management   
UGF Affiliated*Delhi, India

Yash Sharma  
*Dept. Computer Science and Engineering   
UGC Affiliated*  
*HMR Institute of Technology and Management   
UGC Affiliated*Delhi, India Suhit Garg  
*Dept. Computer Science and Engineering   
UGC Affiliated*  
*HMR Institute of Technology and Management,   
UGC Affiliated*Delhi, India

Anil Vats

Assistant Professor  
*Dept. Computer Science and Engineering   
UGC Affiliated*  
*HMR Institute of Technology and Management   
UGC Affiliated*Delhi, India Shivam Goel  
*Dept. Computer Science and Engineering   
UGC Affiliated*  
*HMR Institute of Technology and Management   
UGC Affiliated*Delhi, India

***Abstract- Because of an outstanding development in the age of literary information, the requirement for mechanism and tools for summarizations of textual data has gotten exceptionally critical. Accordingly, a programmed summarizer is indispensable towards diminishing human efforts. Text summarization is a significant movement in the study of high-volume text reports and is presently a major exploration and research theme in Natural Language Processing. It is the technique of producing of the summary of input document by extracting the characteristic sentences from it. In this project, we present a novel strategy for producing the summary of content specific text by utilizing Semantic Analysis for text summarizations a subset of Natural Language Processing.***

*Keywords – NLP, Text summarization, Extractive Summarization.*

1. INTRODUCTION

Text summarization is the making of an abbreviated variant of a text or a document by program. Summarization has become a very helpful way of tackling the issue of data overburden in the 21st century. The result of this method actually contains the most significant points of the base document and is by and large alluded to as a summary.

Comprehensively, one recognizes two ways to deal with text summarization: extraction and abstraction. In Extraction Summarization, information which is computed to be the most important by the program is taken into account from the original text. While, in the case of abstraction summarization, the final summary is produced by paraphrasing of the original text. Abstraction summarisation programs are much more complex and harder to develop in contrast to extraction summarisation programs. natural language processing and/or statistical methods are used in both of the aforementioned techniques. Text Summarization discovers its applications in different NLP related fields like Question Answering, Text Characterization. Summarization program is incorporated into these frameworks as a transitional stage which assists with lessening the length of the document. This, thus, leads to quicker access for data searching.

In this paper, our focus will be on extractive summarization technique. I utilize extractive summary since I can apply this technique to numerous reports without doing a great deal of (overwhelming) AI model training tasks. Other than that, extractive synopsis gives preferably better result over abstractive summarization, in light of the fact that abstractive summarization needs to create new sentences from the source content, which is a more troublesome technique than an information driven way to deal with separate significant sentences. It centers around extracting objects straightforwardly from the entire text without changing the actual articles. A dictionary of word count will be created and on it’s basis, the sentences will be rated. Furthermore, a final sentence rating dictionary will be created and the top sentences will be picked for the resultant summary. The model is not data driven but is independent of the inputted source document. Moreover, PyQt5 has been used for a user-friendly user interface. PyQt is a GUI widgets toolkit. It is a Python interface for Qt, one of the most powerful, and popular cross-platform GUI library. PyQt API is a set of modules containing a large number of classes and functions.

1. LITERATURE REVIEW

Hamza Shabbir et al (2020) proposed the use of Latent Semantic Analysis for summarization. They used their summariser which was derived from Natural Processing Library libraries on the inputted document. Document is processed through step involving semantic rules, dictionaries and SVD(singular value decomposition) after the removal of stop words.

Josef Steinberger et al, described a generic text summarization method. They also used the LSA technique to discover semantically important sentences. They also suggested a couple of more techniques which measures content similarity between an source text and summary

M. S. Patil et al (2014), proposed a summarization framework dependent on a few extractive content summarization methods, and on the Support-Vector Machine(SVM). This framework attempts to improve the quality of the summary produced by the clustering technique by surging it with SVM

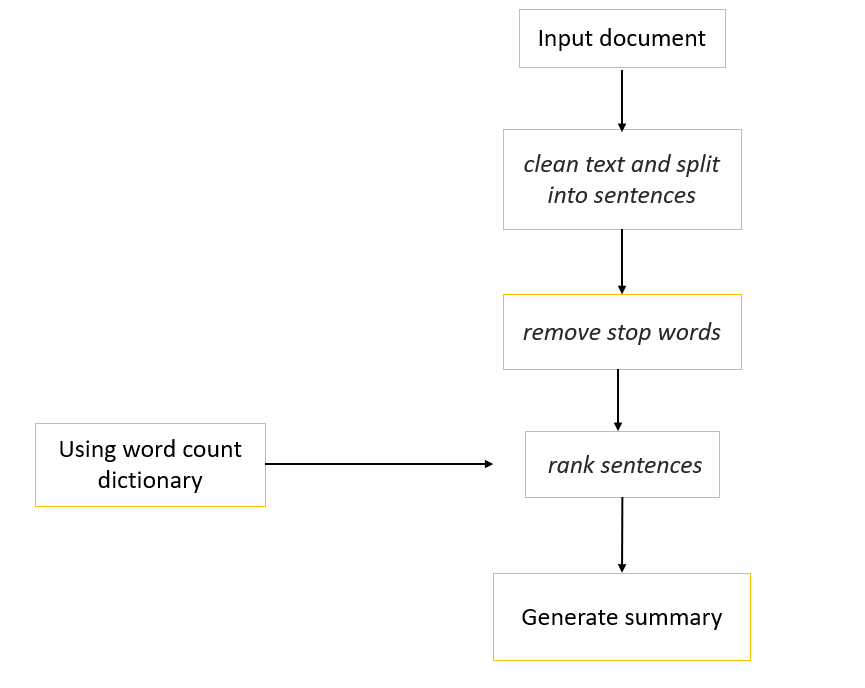
Rasimet et al proposed a framework for summarization utilizing the extractive system and made use of a evolutionary algorithm. In their research, they proposed a unsupervised text summarization strategy that makes the summary by grouping and extricating sentences from the source document.

1. DISCUSSION

According to our research, it is apparent that extractive summarization has had a more success than abstractive summarization. Extractive summarization gives better summary outcome than abstractive summary, because abstractive summarization has to generate new sentences from the original text. Although, if a successful implementation of abstractive summarization takes place, it will be superior to the extractive summarization and will make more sense than the former one.

1. PROPOSED SYSTEM

The proposed system focuses on the aspect of generating summary from the original source document. The workflow of our framework, as defined in figure 1, is carried out in phases. Basic libraries in python are imported. User document is uploaded via a user-friendly interface created by PyQt5. The file format should be a .docx, .pdf or .txt else an error message is flashed on the user interface. Sentences having length over 30 has been neglected due to the fact that longer sentences have a high chance of providing less important information. The document’s text is cleaned and split into sentences. Stop words like “is”, “the”, etcetera, are removed from the sentences for the creation of word count dictionary. The sentences are then ranked on the basis of the word count, that is, sentence score is the cumulative score of the words present in the sentence. Top N sentences will be selected to form the summary, where N is the quarter of the total number of sentences present in the document. The words will be semantically and syntactically logical as they will be selected from the source document. The final summary will can be viewed on the application interface and a separate text file will be created at the same location as of the source file.



*Fig 1:- Flow Chart of Summary Generator*

***A. Source Code***

def push\_button\_Submit(self):

sent\_word\_length = '30'

path = self.file\_name[0]

#file will be inputed

file = self.read\_content(path)

text = file

text = re.sub(r'\[[0-9]\*\]',' ',text)

text = re.sub(r'\s+',' ',text)

clean\_text = text.lower()

regex\_patterns = [r'\W',r'\d',r'\s+']

for regex in regex\_patterns:

clean\_text = re.sub(regex,' ',clean\_text)

sentences = nltk.sent\_tokenize(text)

top\_n= math.sqrt(len(sentences))

stop\_words = nltk.corpus.stopwords.words('english')

word\_count = {}

for word in nltk.word\_tokenize(clean\_text):

if word not in stop\_words:

if word not in word\_count.keys():

word\_count[word] = 1

else:

word\_count[word] += 1

sentence\_score = {}

for sentence in sentences:

for word in nltk.word\_tokenize(sentence.lower()):

if word in word\_count.keys():

if len(sentence.split(' ')) < int(sent\_word\_length):

if sentence not in sentence\_score.keys():

sentence\_score[sentence] = word\_count[word]

else:

sentence\_score[sentence] += word\_count[word]

best\_sentences = heapq.nlargest(int(top\_n), sentence\_score, key=sentence\_score.get)

summarized\_text = []

sentences = nltk.sent\_tokenize(text)

for sentence in sentences:

if sentence in best\_sentences:

summarized\_text.append(sentence)

summarized\_text = "\n".join(summarized\_text)

self.lblSubmit.setText(summarized\_text)

savepath=self.path.split('/')

savepath=savepath[:-1]

savepath='/'.join(savepath)+'/'+'summary.txt'

text\_file = open(savepath, "w")

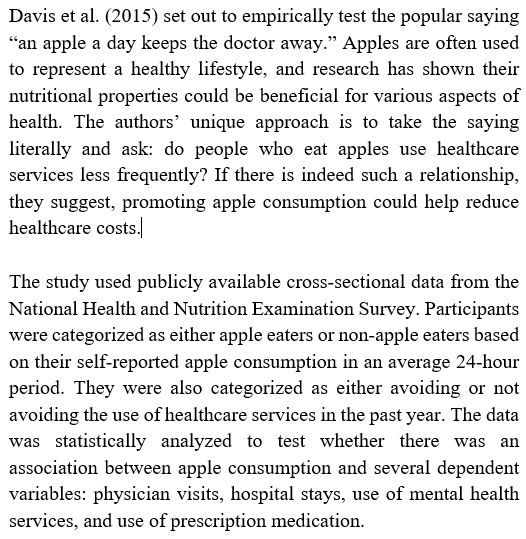
text\_file.write(summarized\_text)

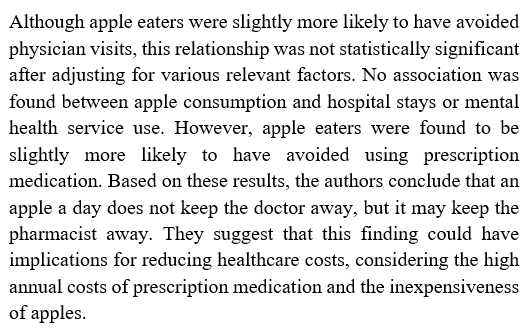
text\_file.close()

1. RESULTS

***A. Observations***

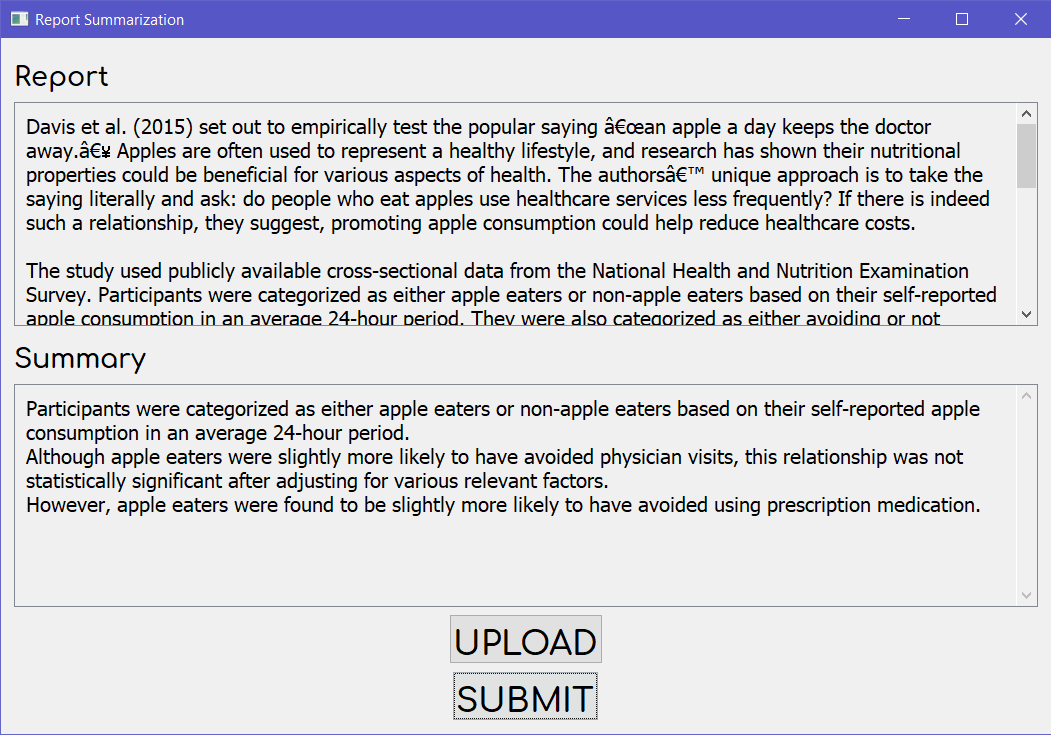
* This summary generator can summarize files of any size.
* Summary will be exactly 1/4th of the actual file.
* Summary formation is real quick function of the following program. It can convert huge reports, files, text files, paragraphs, essays, comprehensions within seconds.
* It is very simple to under stand and any one with zero technical knowledge can access it easily. As one just has to upload the file in the report section. And just has to click submit to generate the summary.





*Fig 1 & 2:- Source Document*

The results of our summary generator using the summarization technique is shown in this section.



*Fig 4:- UI & Summary*

1. CONCLUSION AND FUTURE SCOPE

Text summarization is one of the critical problems in the field of Natural Language Processing. It is important to note that we used word frequency in a document to rank the sentences. The advantage of using this method is that it does not require any prior training and can work on any piece of text. We can further change the summarizer by changing the either or both of the parameters namely the number of top sentences and the sentence length. We have set this parameter in accordance to the inferences which can be drawn from the English language.

1. REFERENCES
2. Hamza Shabbir Moiyadi, Harsh Desai, Dhairya Pawar, Geet Agrawal, Nilesh M.Patil, "NLP Based Text Summarization Using Semantic Analysis", International Journal of Advanced Engineering, Management and Science (IJAEMS)
3. Josef Steinberger, KarelJežek, “Using latent Semantic analysis In Text Summarization and Summary Evaluation”, Department of Computer Science and Engineering, UniverzitníCZ-306 14 Plzeň.
4. M. S. Patil, M. S. Bewoor, S. H. Patil “A Hybrid Approach for Extractive Document Summarization Using Machine Learning and Clustering Technique.”International Journal of Computer Science and Information Technologies, Vol. 5, IssueNo. 2, ISSN: 0975-9646, pp.1584-1586, 2014
5. RasimAlguliev, RamizAliguliyev, “Evolutionary Algorithm for Extractive Text Summarization.” Intelligent Information Management, 1, pp. 128-138, November 2009. [6] M. S. Patil, M. S. Bew
6. Michael Ji, “Text Summarization Tool Evaluation: A Feasibility Study for Generating Meeting Summaries.” CPSC503 Final Report, Department of Computer Science, University of Calgary.
7. Anne HendrikBuist, Wessel Kraaij and Stephan Raaijmakers, “Automatic Summarization of Meeting Data: A Feasibility Study.”
8. Ronan Collobertcollober, Jason Weston, “A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning.” ]Edward Hovy, Chin-Yew Lin, “Automated Text Summarization and the Summarist System”, Information Sciences Institute of the University of Southern California.
9. Jen-Yuan Yeh, Hao-RenKe, Wei-Pang Yang, I HengMeng, “Text summarization using a trainable summarizer andlatent semantic analysis”, Elsevier Proceeding of Information processing and management