

SRM UNIVERSITY

TEXT SUMMARIZER

15IT375L- MINOR PROJECT REPORT

Submitted by

SREE PADMAVATHY.B (RA1711008010001)

AISHWARYA .S (RA1711008010012)

SHIRAN ANAND (RA1711008010025)

AVIRAL KATIYAR (RA1711008010064)

for the assessment of 3rd year Minor Project

in the

DEPARTMENT OF INFORMATION TECHNOLOGY



SRM UNIVERSITY

KATTANKULATHUR

MAY,2020

SRM UNIVERSITY

KATTANKULATHUR

BONAFIDE CERTIFICATE

Certified that this Minor project report “**TEXT SUMMARIZER**” is the bonafide work of “**SREE PADMAVATHY B,AISHWARYA S,SHIRAN ANAND,AVIRAL KATTIYAR**” who carried out the project work under my supervision at SRM University , IT Department, Kattankulathur.

SIGNATURE

Guide Name

SIGNATURE

HEAD OF THE DEPARTMENT
Information Technology

INTERNAL EXAMINER

Evaluation Sheet
TEXT SUMMARIZER

Register Number:

Name:

Description	Marks	Marks Obtained
Project Report	30	
Research Paper	10	

Minor Project Coordinator,

ACKNOWLEDGEMENT

The success and the final outcome of this project required guidance and assistance from different sources and we feel extremely fortunate to have got this all along the completion of our project. Whatever we have done is largely due to such guidance and assistance and we would not forget to thank them.

We express our sincere thanks to the Head of the Department, Department of Information Technology, Dr.G.Vadivu, for all the help and infrastructure provided to us to complete this project successfully and his valuable guidance.

We owe our profound gratitude to our project guide Mrs **SINDHU S**, who took keen interest in our project work and guided us all along, till the completion of our project work by providing all the necessary information for developing a good system.

We are thankful to and fortunate enough to get constant encouragement, support and guidance from all the Teaching staff of the Department of Information Technology which helped us in successfully completing our minor project work. Also, we would like to extend our sincere regards to all the non-teaching staff of the department of Information Technology for their timely support.

SREE PADMAVATHY B,AISHWARYA S,SHIRAN ANAND,AVIRAL KATIYRA
RA1711008010001,RA1711008010012,RA1711008010025,RA1711008010064

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	6
1	INTRODUCTION	7
2	REQUIREMENT ANALYSIS WORK	10
3	DESIGN	13
4	IMPLEMENTATION	18
5	TESTING	28
6	CONCLUSION	35
7	REFERENCES	36

ABSTRACT

In this new epoch, where colossal information is available on the internet, it is difficult to extract the required information quickly and efficiently since there are plenty of information available on the internet. So there is a problem of searching for relevant documents from the number of documents available, and absorbing relevant information from it. Summarization of text is necessary in order to find relevant information.

The Text summarization tool is used to summarize the articles thereby saving the individual's time and resources. The goal of this Project is to create a text summarization tool which can help summarize documents , articles and other passages by feeding it with data/datasets. Following the goal , we developed an NLP model for text summarization and trained it on datasets.

CHAPTER 1

INTRODUCTION

Text Summarizer revolves around the concept of Natural Language Processing(NLP) NLP is broadly defined as the automatic manipulation of natural language, like speech and text, by software.

The study of language process has been around for over fifty years and grew out of the sphere of linguistics with the increase of computers. Natural language refers to the way we, humans, communicate with each other. Namely, speech and text. We are surrounded by text and speech. Voice and text are how we communicate with each other.

Natural language processing is a part of artificial intelligence. By utilizing NLP, developers can organize and structure knowledge to perform tasks such as automatic summarization, translation, named entity recognition, relationship extraction, sentiment analysis, speech recognition, and topic segmentation.

NLP is used to analyze text, allowing machines to understand how human's speak.

This human-computer interaction enables real-world applications like automatic text summarization, sentiment analysis, topic extraction, named entity recognition, parts-of-speech tagging, relationship extraction, stemming, and more. NLP is commonly used for text mining, machine translation, and automated question answering.

OVERVIEW

Automatic text summarization is the task of producing a concise and fluent summary while preserving key information content and overall meaning.

The project consists of a web application using react for the user communication.

The user is asked to enter the text or URL. The text is then summarized using Natural Language Processing techniques. The summarized text is then made visible to the user. Extractive text summarization method is used to capture the summary of the source document. Extractive summarization means identifying important sections of the text and generating them verbatim producing a subset of the sentences from the original text

OBJECTIVE

Text summarizer is employed for making a brief and correct outline from an extended text or address. Summarization ways are greatly required to consume the ever-growing quantity of text knowledge accessible on-line. In essence, summarization is supposed to assist the user consume relevant data in a short period of time. The main objective of a text summarization system is to spot the foremost vital data from the given text and gift it to the tip users.

Theoretic text summarisation is bestowed by distinguishing text options and grading the sentences consequently. The text is initial pre-processed to tokenize the sentences and perform stemming operations. Sentences are then processed using the different text features. Two novel approaches implemented are using the citations present in the text and identifying synonyms. The summarized content is then displayed to the end user.

The objective is to provide a brief summary of the text and remove manual work as well as save a lot of time and resource.

CHAPTER 2

REQUIREMENT ANALYSIS WORK

LITERATURE STUDY

Numerous approaches for identifying important content for automatic text summarization have been developed to date. Topic representation approaches first derive an intermediate representation of the text that captures the topics discussed in the input. Based on these representations of topics, sentences in the input document are scored for importance. In contrast, in indicator representation approaches, the text is represented by a diverse set of possible indicators of importance which do not aim at discovering topicality. These indicators are combined, very often using machine learning techniques, to score the importance of each sentence. Finally, a summary is produced by selecting sentences in a greedy approach, choosing the sentences that will go in the summary one by one, or globally optimizing the selection, choosing the best set of sentences to form a summary. In this chapter we give a broad overview of existing approaches based on these distinctions, with particular attention on how representation, sentence scoring or summary selection strategies alter the overall performance of the summarizer. We also point out some of the peculiarities of the task of summarization which have posed challenges to machine learning approaches for the problem, and some of the suggested solutions.

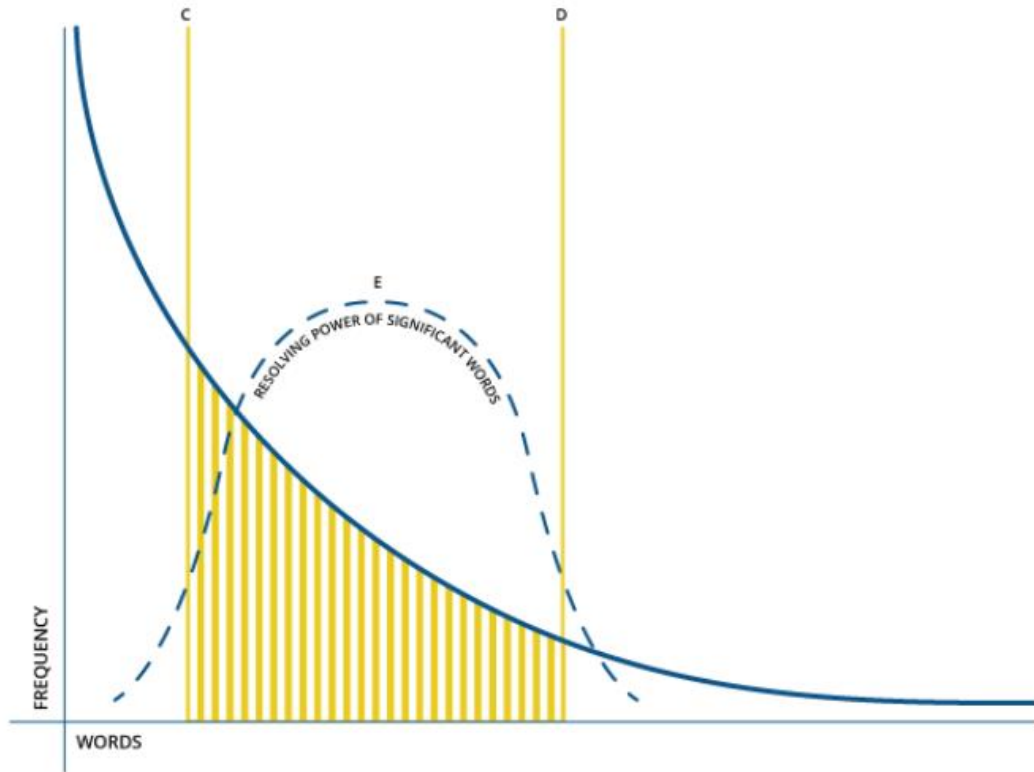
The importance of the sentences is decided based on statistical and linguistic features of sentences. An abstractive summarization is used to understanding the main concepts in a given document and then expresses those concepts in clear natural language. In this paper, gives comparative study of various text summarization techniques.

LITERATURE REVIEW

Author	Year	Techniques/Methods	Outcome
Chin-Yew Lin <i>et al.</i>	2004	Graph based approach	Abstractive Summary generation of redundant data
Akshil Kumar <i>et al.</i>	2017	Graph based approach, Semantic based approach	Performance of Three different algorithms compared TextRank, LexRank and LSA. TextRank outperforms other two.
Pankaj Gupta <i>et al.</i>	2016	Sentiment Analysis, Text Summarization Techniques	A Survey is performed on current research in sentiment analysis and Text summarization.
Harsha Dave <i>et al.</i>	2015	Ontology based	Generated abstractive summary from extractive summary
Yihong Gong <i>et al.</i>	2001	Semantic based	New LSA method provides generic text summary.
Rada Mihalcea <i>et al.</i>	2004	Graph based	New TextRank method generates extractive text summary.
Güneş Erkan <i>et al.</i>	2004	Graph based	New LexRank method generates extractive text summary
Kavita Ganesan <i>et al.</i>	2010	Graph based	New framework Opinosis generates abstractive summary of redundant data
Dharmendra Hinhu <i>et al.</i>	2015	Extractive Text summarization approach	Extractive text summarization approach is used to summarize Wikipedia Articles.
N. Moratanchet <i>et al.</i>	2016	Structure based, Semantic based Approaches	A Survey on various techniques of Abstractive text summarization.
Tacho Jo	2017	K- Nearest Neighbor	Modified KNN provides Text summarization

REQUIREMENT ANALYSIS

An analytic method to extract salient sentences from the text using features such as *word and phrase frequency*. In this method sentences of the document are weighed as a function of high frequency words, ignoring very high frequency common words this approach that became the one of the pillars of NLP.



CITATION ANALYSIS

The basic behaviors of general summarization include emerging, selecting, citing and organizing representations at language level, concept level and knowledge level according to requirement and motivation. Text summarization is a special case of general summarization.

The summarization S of a set of representations P is defined by its intension $\text{Intension}(P)$ and extension $\text{Extension}(P)$ as follows, where $S(\text{Intension}(P))$ is the summarization of the intension of P , $S(\text{Intension}(P) \cup \text{Extension}(P))$ is the summarization of the union of the intension of P and the extension of P , $p \rightarrow p'$ means that p cites p' , $\{\text{Cite}(p_i \rightarrow p) \mid i \in [1, \dots, m]\}$ denotes the set of cite representations in p_i that cites p , and $\text{Cite}(p_i \rightarrow p)$ describes p from the view of the author of p_i . The union refers to set union or graph union if the intension and extension take the form of sets or graphs.

$$S(P) = \langle S(\text{Intension}(P)), S(\text{Intension}(P) \cup \text{Extension}(P)) \rangle.$$

$$\text{Intension}(P) = \{ \langle \text{Intension}(p), \dots, \text{Intension}(p) \rangle \mid p \in P \}.$$

$$\text{Intension}(p) = \{ \langle \text{Core}(p), \text{Core}(p) \cup \text{Cite}(o_k \rightarrow p) \mid k \in [1, \dots, n], p \in P, o_k \in P \rangle \}.$$

$$\text{Extension}(P) = \{ \cup \text{Extension}(p) \mid p \in P \}.$$

$$\text{Extension}(p) = \{ \text{Intension}(q_i) \cup \text{Intension}(o_j) \mid p \rightarrow q_i, o_j \rightarrow p, i, j \in [1, \dots, n], p \in P, q_i \in P, o_j \in P \}.$$

Different from the previous notions of summarization, this definition gives the minimum summary (the summary of the intension) and the maximum summary (the summary of the intension and the extension) of a representation, and it regards citation as the fundamental behavior and mechanism of summarization.

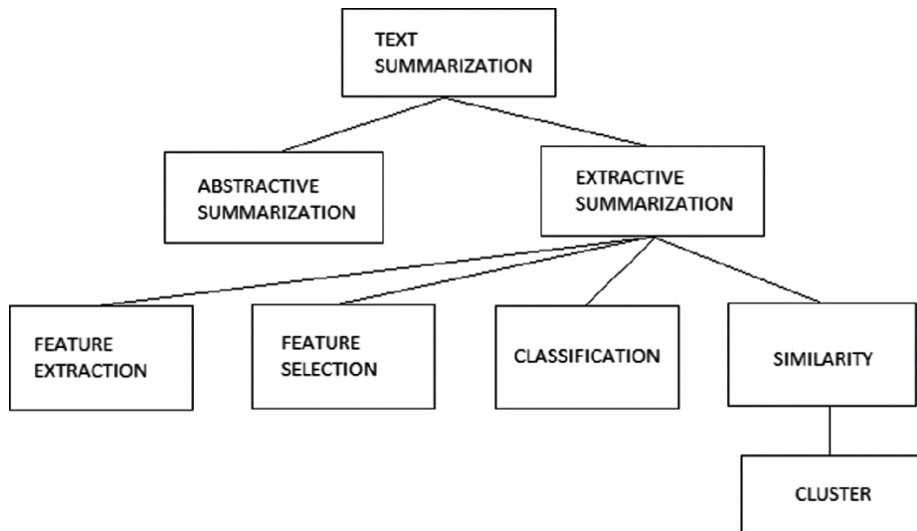
CONCLUSION

As the availability of data in the form of text increasing day by day. It becomes so difficult to read the whole textual data in order to find the required information which is both difficult as well as a time-consuming task for a human being. So, at that time ATS performs an important role by providing a summary of a whole text document by extracting only the useful information and sentences. There are different approaches of text summarization. The real-world applications of text summarization can be: documents summarization, news and articles summarization, review systems, recommendation systems, social media monitoring, survey responses systems. The paper provides a literature review of various research works in the field of automatic text summarization. This research area can be explored more by looking in existing systems and working on different and new techniques of NPL and Machine Learning.

CHAPTER 3

DESIGN

GENERAL ARCHITECTURE FOR TEXT SUMMARIZER



PROPOSED SYSTEM

Text summarization is a subdomain of Natural Language Processing (NLP) that deals with extracting summaries from huge chunks of texts.

we will see a simple NLP-based technique for text summarization.

we will be using Python's NLTK library for summarizing the articles/datasets.

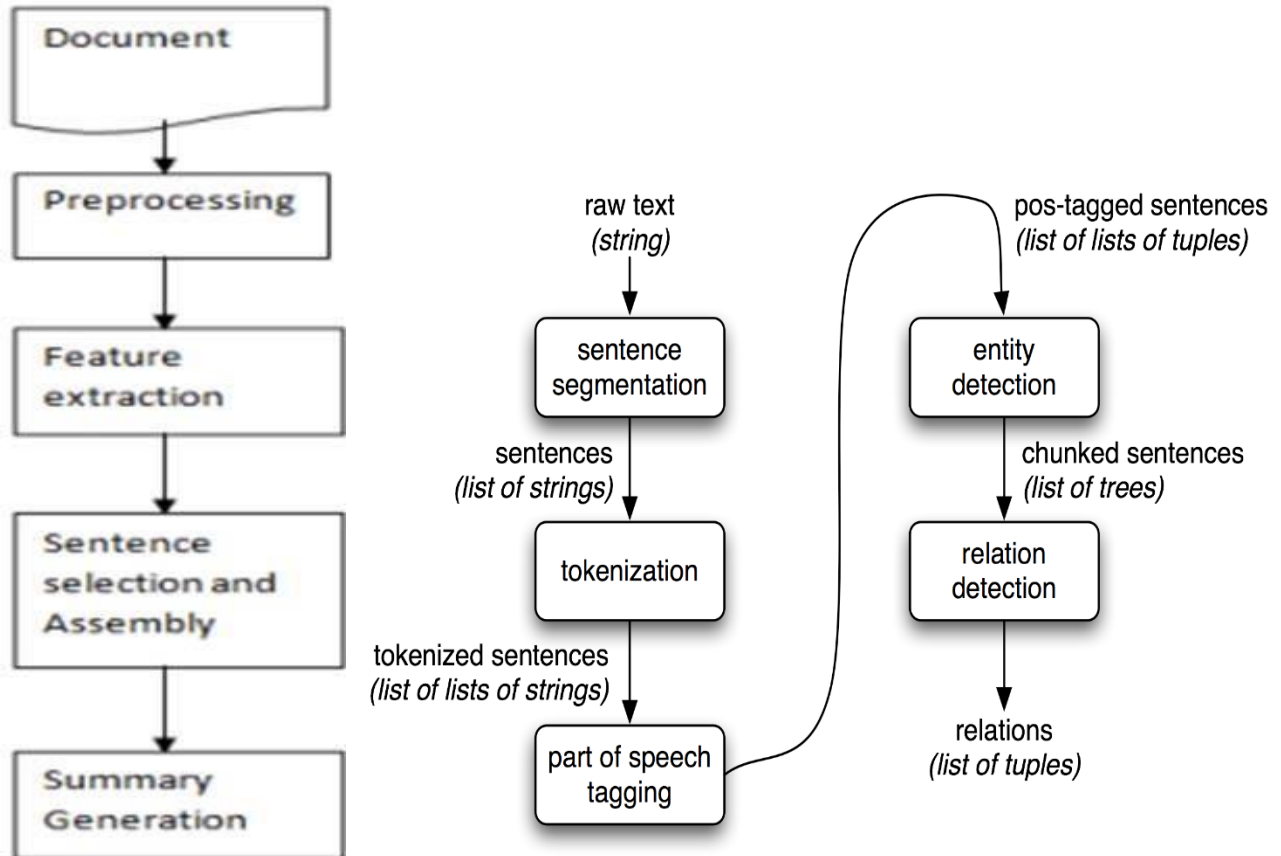
we will be focusing on the extractive summarization technique.

Extractive Summarization: These methods rely on extracting several parts, such as phrases and sentences, from a piece of text and stack them together to create a summary. Therefore, identifying the right sentences for summarization is of utmost importance in an extractive method.

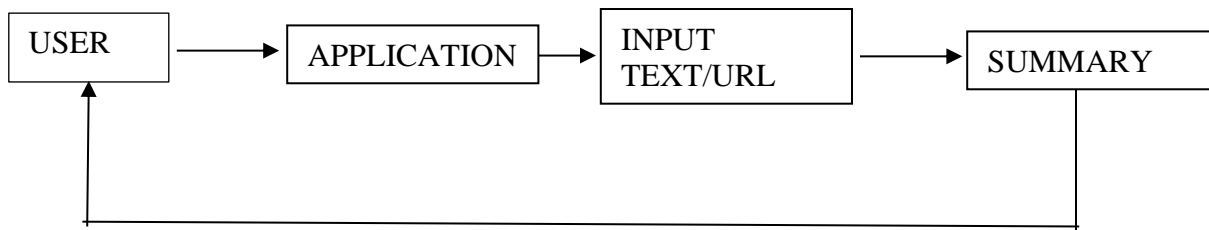
For example, The mobile app called “**inshorts**” is an innovative news app that converts news articles into a 60-word summary. It uses TextRank algorithm to process the resources. The process is named as Automatic Text Summarization.

It is a process of generating a concise and meaningful summary of text from multiple text resources such as books, news articles, blog posts, research papers, emails, and tweets.

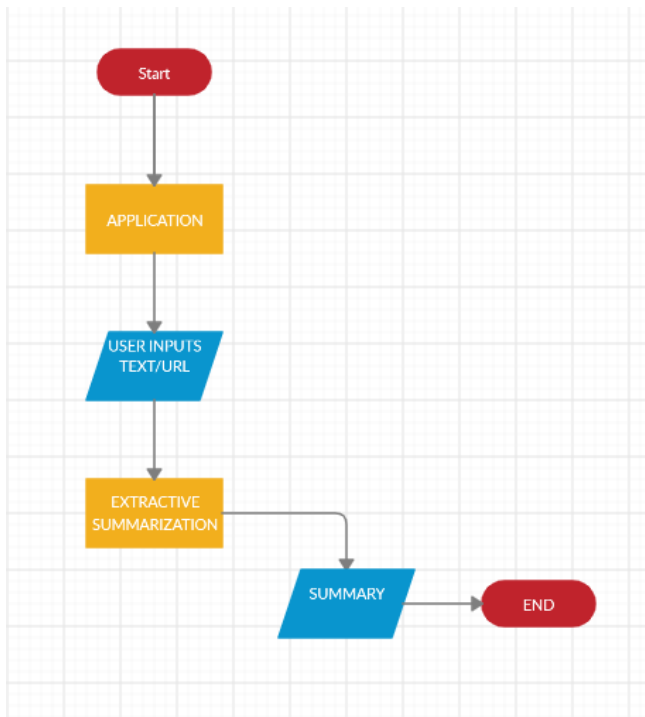
ARCHITECTURE DIAGRAM



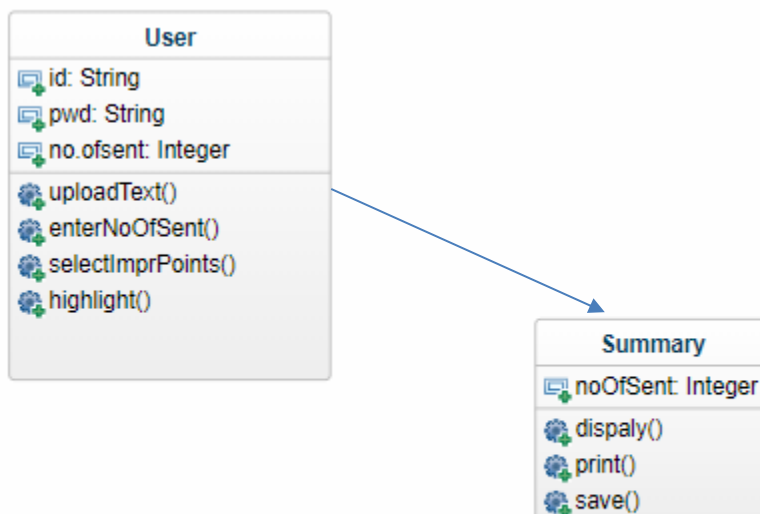
SYSTEM DESIGN



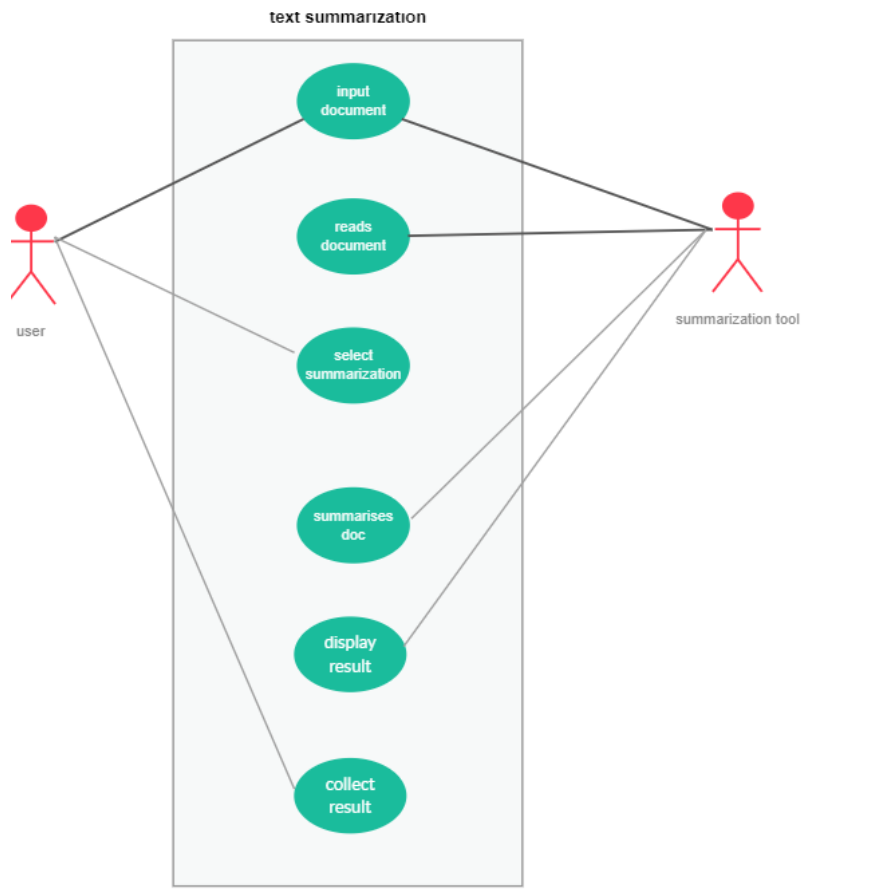
FLOW CHART



CLASS DIAGRAM



USE CASE DIAGRAM



CHAPTER 4

IMPLEMENTATION

LIBRARIES USED

NLTK

NLTK is a leading platform for building Python programs to figure with human language knowledge. It provides easy-to-use interfaces to over fifty corpora and lexical resources like WordNet, along side a collection of text process libraries for classification, tokenization, stemming, tagging, parsing, and linguistics reasoning, wrappers for strong human language technology libraries, and a full of life discussion forum.

NEWSPAPER3K

Newspaper is a Python 2 library for extracting & curating articles from the web. It wants to change the way people handle article extraction with a new, more precise layer of abstraction.

SOURCE CODE

CLIENT SIDE SCRIPTING

```
App.js

import React from 'react';
import './App.css';
import "bootstrap/dist/css/bootstrap.min.css";
import { Route, BrowserRouter as Router, Switch } from "react-router-dom";
import textSum from "./textSum";
import urlText from "./urlText";
import intro from "./intro";

function App() {

  return (
    <div>
      <Router>
        <Switch>
          <Route exact path="/" component={intro} />
          <Route path="/textSum" component={textSum} />
          <Route path="/urlText" component={urlText} />
        </Switch>
      </Router>
    </div>
  );
}
```

```

    </Router>
  </div>
);
}

export default App;

2. intro.js

import React, { Component } from 'react';
import './App.css';
import 'bootstrap/dist/css/bootstrap.min.css';
import { Button, Col, Form, FormGroup } from 'react-bootstrap';

class intro extends Component {
  textSum() {
    let path = '/textSum';
    this.props.history.push(path);
  }
  urlText() {
    let path = '/urlText';
    this.props.history.push(path);
  }

  render() {
    return(
      <div>
        <header>
          <div className="head-intro">
            <h2>Text Summarizer using NLP</h2>
          </div>
          <div className="button">
            <div className="but-ind">
              <Button onClick={this.textSum.bind(this)} variant="secondary">
                Text Summarizer
              </Button>
            </div>
            <div className="but-ind">
              <Button onClick={this.urlText.bind(this)} variant="secondary">
                URL Text Summarizer
              </Button>
            </div>
          </div>
        </header>
      </div>
    );
  }
}

```

```

        </div>
    );
}
}

export default intro;
3. textSum.js (For user input text):

import React, { Component } from 'react';
import "bootstrap/dist/css/bootstrap.min.css";
import { Button, Col, Form, FormGroup } from "react-bootstrap";
import './App.css';
import axios from "axios";

class textSum extends Component {
  constructor(props) {
    super(props);
    this.state = {
      textarea: "",
    };
  }

  getResult(data) {
    alert(data.textarea);
    const dataurl = "http://127.0.0.1:5000/textSum";
    return axios.post(dataurl, data);
  }

  Sendtext = e => {
    e.preventDefault();
    debugger;
    alert("ok");
    let obj = {
      textarea: this.state.textarea
    };
    this.getResult(obj).then(res => {
      this.setState({ message: "Yes" });
      // this.props.history.push("/UserDash");
      if (res.data.results[0] !== "") {
        alert("Summarized");
        let inner_h = document.querySelector(".inner");
        inner_h.textContent = res.data.results;
      }
      else {
        alert("Summarized");
        let inner_h = document.querySelector(".inner");

```

```

        inner_h.textContent = res.data.results[0];
    }
    });
};
handleChange = e => {
    this.setState({ [e.target.name]: e.target.value });
};

render()
{
    return(
        <div>
        <div className="head-intro">
            <h2>Input Text Summarizer</h2>
        </div>
        <Form>
            <Form.Group controlId="textarea">
                <Form.Label className="form-
text">Enter the text to be summarized</Form.Label>
                <Form.Control as="textarea" rows="8"
                    name="textarea"
                    value={this.state.textarea}
                    onChange={this.handleChange} />
            </Form.Group>
            <Button className="but-
form" onClick={this.Sendtext.bind(this)} variant="secondary" type="submit">
                Submit
            </Button>
        </Form>
        <div className="inner"></div>
    </div>
    );
}
}

```

export default textSum;

4. urltext.js (For summary of articles in a webpage):

```

import React, { Component } from 'react';
import "bootstrap/dist/css/bootstrap.min.css";
import { Button, Col, Form, FormGroup } from "react-bootstrap";
import './App.css';
import axios from "axios";
class urlText extends Component {
    constructor(props) {

```

```

    super(props);
    this.state = {
      urltext: "",
    };
  }

  getResult(data) {
    alert(data.urltext);
    const dataurl = "http://127.0.0.1:5000/urltext";
    return axios.post(dataurl, data);
  }

  Sendtext = e => {
    e.preventDefault();
    debugger;
    alert("ok");
    let obj = {
      urltext: this.state.urltext
    };
    this.getResult(obj).then(res => {
      this.setState({ message: "Yes" });
      // this.props.history.push("/UserDash");
      if (res.data.results[0] !== "") {
        alert("Summarized");
        let inner_h = document.querySelector(".inner");
        inner_h.textContent = res.data.results;
      }
      else {
        alert("Summarized");
        let inner_h = document.querySelector(".inner");
        inner_h.textContent = res.data.results[0];
      }
    });
  };

  handleChange = e => {
    this.setState({ [e.target.name]: e.target.value });
  };

  render()
  {
    return(
      <div>
        <div className="head-intro">
          <h2>URL Text Summarizer</h2>
        </div>

```

```

        <Form>
        <Form.Group controlId="urltext">
        <Form.Label className="form-
text">Please Enter the URL here</Form.Label>
        <Form.Control
            name="urltext"
            value={this.state.Sendtext}
            onChange={this.handleChange}
            type="text"
            placeholder="URL"
        />
        </Form.Group>
        <Button className="but-
form" onClick={this.Sendtext.bind(this)} variant="secondary" type="submit">
            Submit
        </Button>
        </Form>
        <div className="inner"></div>
    </div>
    );
}

}

export default urlText;

```

CSS

APP.CSS

```

@import
url('https://fonts.googleapis.com/css2?family=Dancing+Script:wght@562;600;700&di
splay=swap');

```

```

body {
    background-image: url('image.jpg');
    background-repeat: no-repeat;
    background-attachment: fixed;
    background-size: cover;
    background-position: center;
}
.inner {
    color:white;
}

```

```
}  
.head-intro {  
  color:#eee;  
  text-align:center;  
  margin-top: 40px;  
  font-family: 'Dancing Script', cursive;  
  ;  
}
```

```
.form-text {  
  font-family: 'Dancing Script', cursive;  
  color:#eee;  
  font-size: 22px;  
}
```

```
.but-form {  
  margin-left: 625px;  
}  
.button {  
  margin: 60px auto auto 400px;  
  display:flex;  
  background:rgba(0,0,0,0.5);  
  width:40%;  
  height: 200px;  
  border-radius: 5px;  
  border:2px solid #ddd;  
}
```

```
.but-ind {  
  margin: auto;  
  padding: 6px;  
}
```

```
h1 {  
  color:yellow;  
}  
Button {  
  margin:auto auto;  
  border-radius:3px;  
}
```

```

.App {
  text-align: center;
}

.App-logo {
  height: 40vmin;
  pointer-events: none;
}

@media (prefers-reduced-motion: no-preference) {
  .App-logo {
    animation: App-logo-spin infinite 20s linear;
  }
}

.App-header {
  background-color: #282c34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}

.App-link {
  color: #61dafb;
}

@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}

```


SERVER SIDE SCRIPTING

```
import warnings
import pandas as pd
import numpy as np
from flask_cors import CORS
from pandas.io.json import json_normalize
from flask import Flask, request, jsonify
from flask import flash, request
from datetime import date
import sys
import json
from newspaper import Article

from string import punctuation
from nltk.stem.snowball import SnowballStemmer
import nltk

from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import stopwords
from collections import defaultdict
from string import punctuation
from heapq import nlargest

class TextSummarizer:
    #ps = PorterStemmer()
    stemmer = SnowballStemmer("english")
    stopWords = set(stopwords.words("english")+ list(punctuation))
    text = ""
    sentences = ""
    def tokenize_sentence(self):
        words = word_tokenize(self.text)
        print(words)
        return words;

    def input_text(self, t):

        while True:
            self.text = t
            break;
```

```
def cal_freq(self, words):
```

```
    # Second, we create a dictionary for the word frequency table.
```

```
    freqTable = dict()
```

```
    for word in words:
```

```
        word = word.lower()
```

```
        if word in self.stopWords:
```

```
            continue
```

```
        #word = stemmer.stem(word)
```

```
        if word in freqTable:
```

```
            freqTable[word] += 1
```

```
        else:
```

```
            freqTable[word] = 1
```

```
    return freqTable;
```

```
def compute_sentence(self, freqTable):
```

```
    self.sentences = sent_tokenize(self.text)
```

```
    sentenceValue = dict() # dict() creates the dictionary with key and it's  
    corresponding value
```

```
    for sentence in self.sentences:
```

```
        for index, wordValue in enumerate(freqTable, start=1):
```

```
            if wordValue in sentence.lower(): # index[0] return word
```

```
                if sentence in sentenceValue:
```

```
                    sentenceValue[sentence] += index # index return value of  
occurrence of that word
```

```
                    #sentenceValue.update({sentence: index})
```

```
                    #print(sentenceValue)
```

```
                else:
```

```
                    # sentenceValue[sentence] = wordValue
```

```

        sentenceValue[sentence] = index
        #print(sentenceValue)

    print(sentenceValue)
    return sentenceValue;

def sumAvg(self,sentenceValue):
    sumValues = 0
    for sentence in sentenceValue:

        sumValues += sentenceValue[sentence]

    # Average value of a sentence from original text
    average = int(sumValues / len(sentenceValue))

    return average;

def print_summary(self,sentenceValue,average):
    summary = "
    for sentence in self.sentences:
        if (sentence in sentenceValue) and (sentenceValue[sentence] > (1.5 *
average)):
            summary += " " + sentence

    #print(summary)
    return summary

app = Flask(__name__)
CORS(app)

@app.route('/textSum', methods=['POST'])
def text_func():
    data = request.get_json(force=True)
    convert = json_normalize(data)
    s_text = convert.textarea[0]
    ts = TextSummarizer()
    ts.input_text(s_text)

```

```

words = ts.tokenize_sentence()
freqTable = ts.cal_freq(words)
sentenceValue = ts.compute_sentence(freqTable)
avg = ts.sumAvg(sentenceValue)
summary = ts.print_summary(sentenceValue,avg)
d = { }
d["convert"] = summary
dict_j = json.dumps(d)
return jsonify(results=dict_j)

@app.route('/urltext', methods=['POST'])
def url_func():
    data = request.get_json(force=True)
    convert = json_normalize(data)
    url = convert.urltext[0]
    inp = url
    article = Article(inp)
    article.download()
    article.parse()
    article.nlp()
    sum_url = article.summary
    d1 = { }
    d1["convert"] = sum_url
    dict_j = json.dumps(d1)
    return jsonify(results=dict_j)

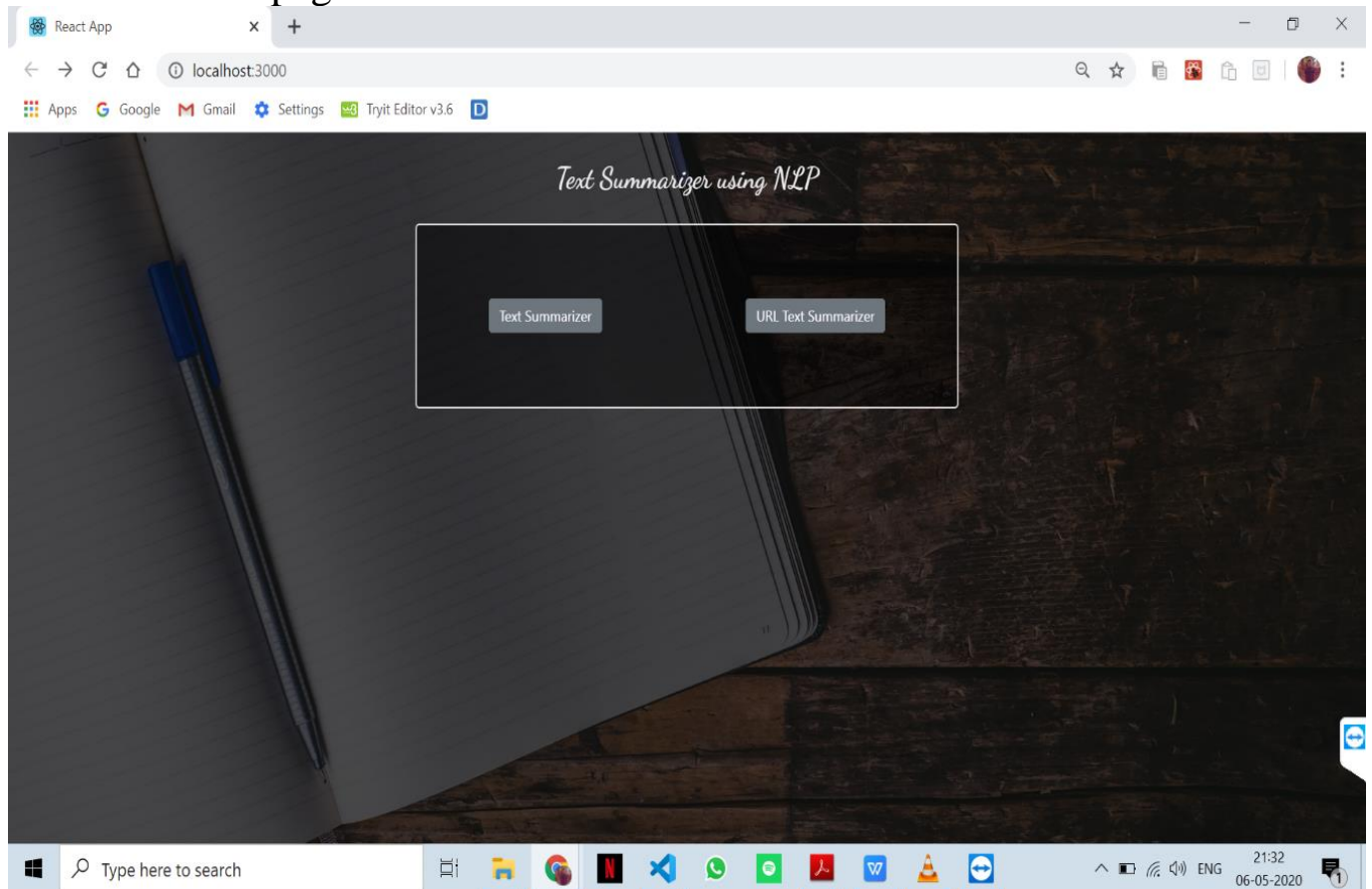
if __name__ == '__main__':
    app.run(debug=True)

```

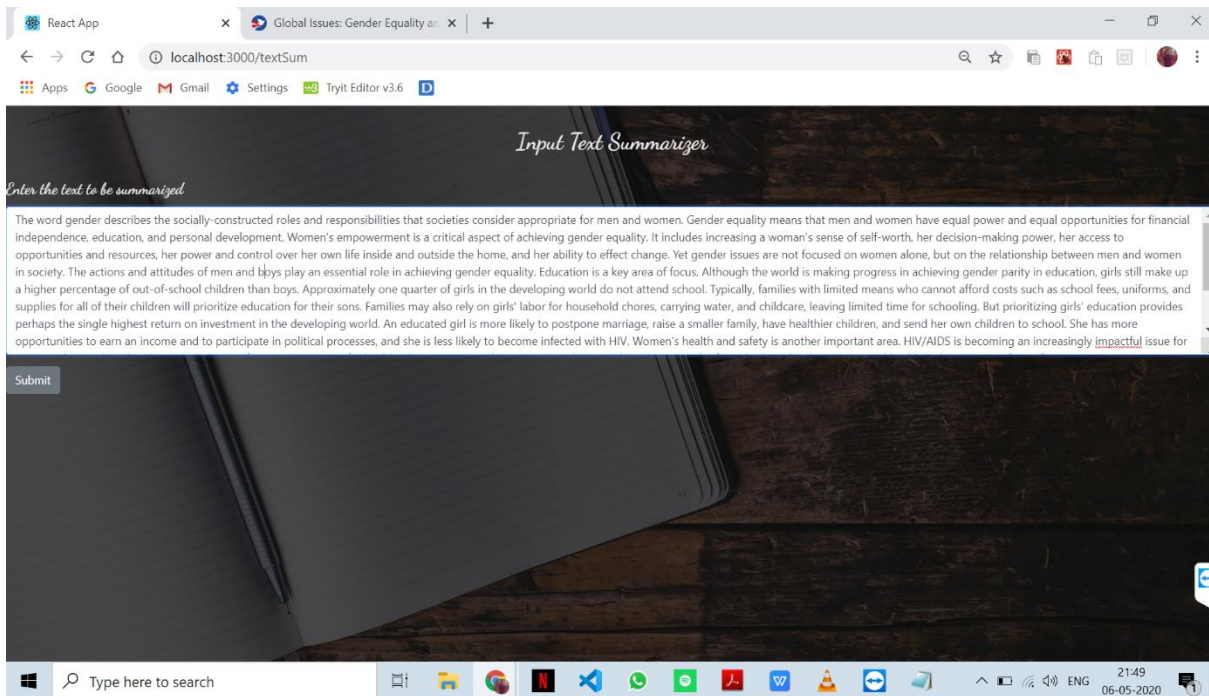
CHAPTER 5 TESTING

SCREENSHOTS OF PROJECT

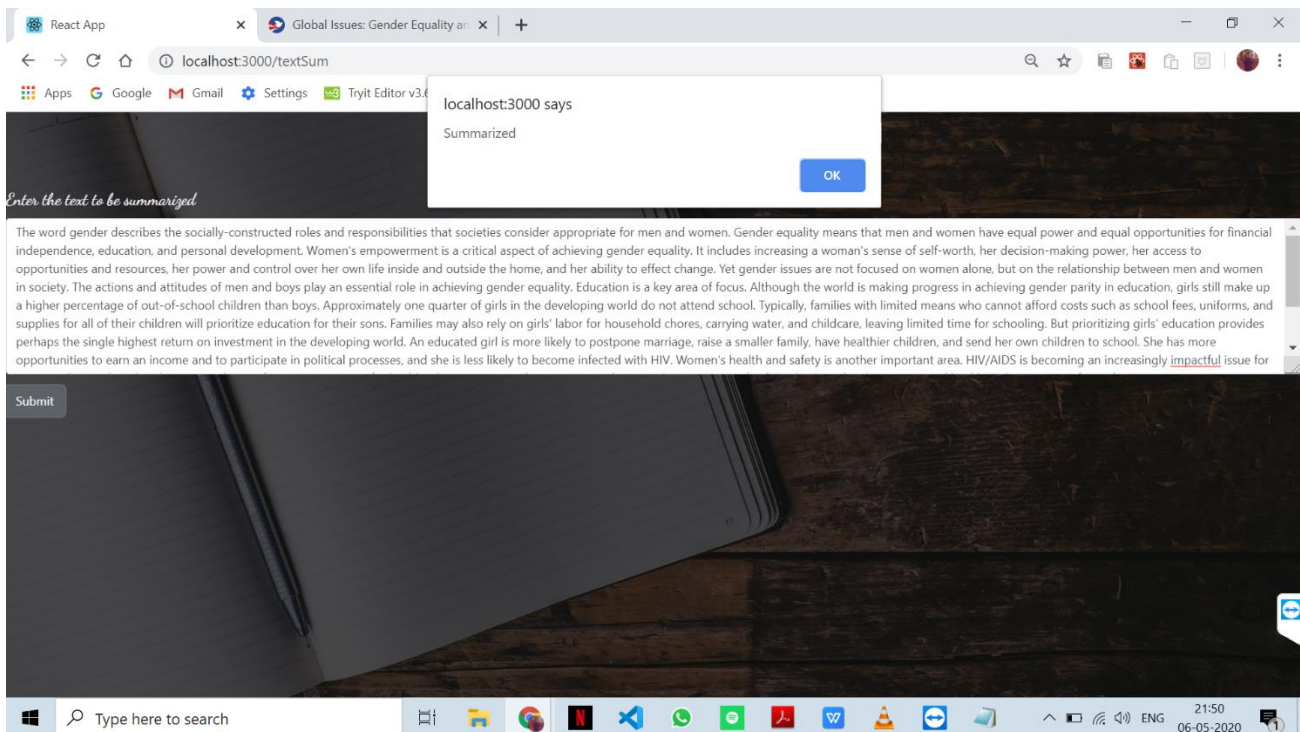
User side front page



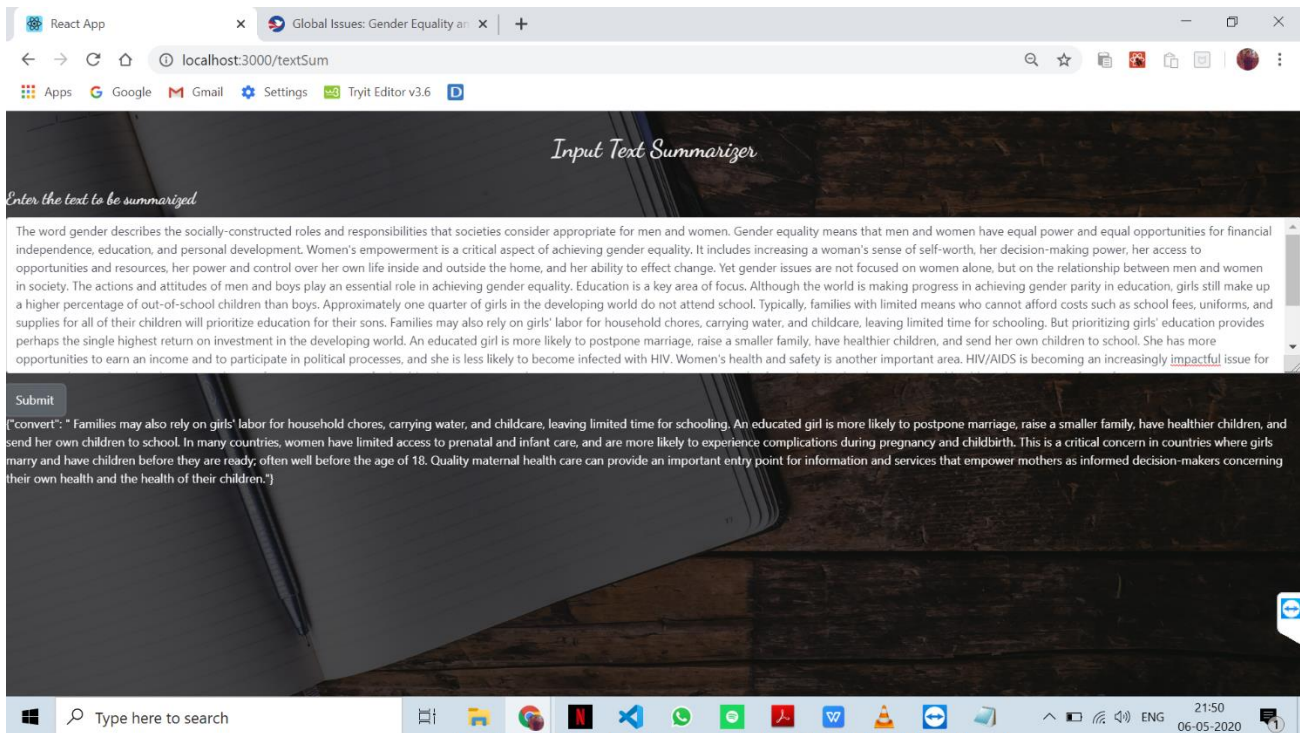
Input Text Summarizer(User enters the data)



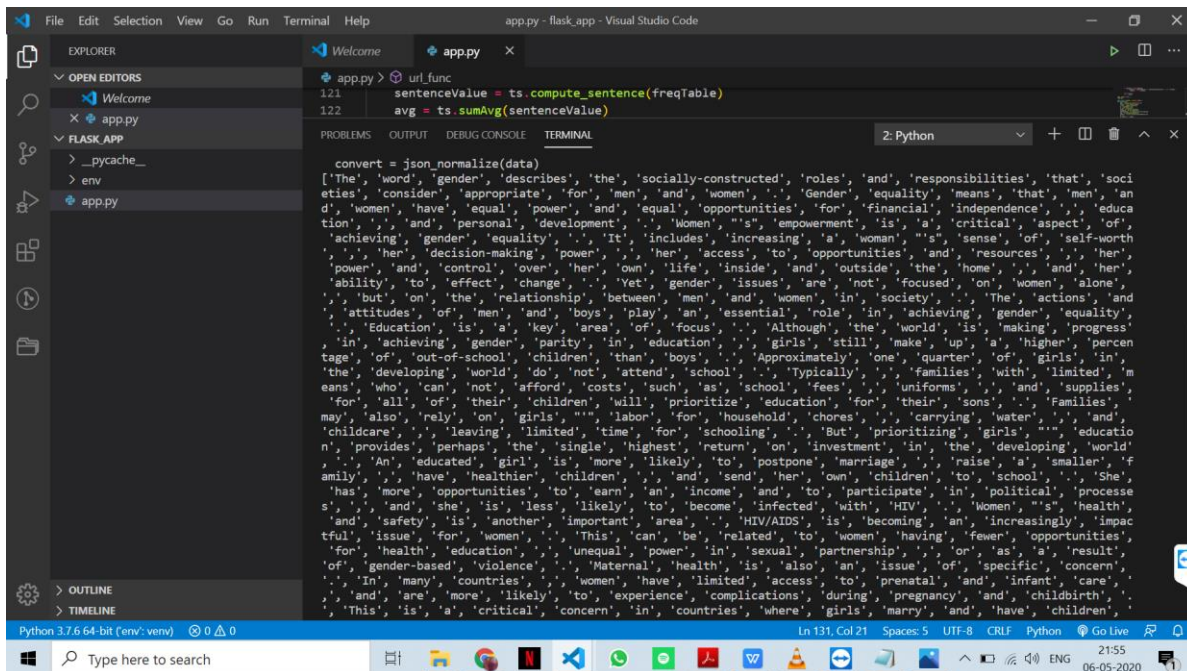
The input is Summarised



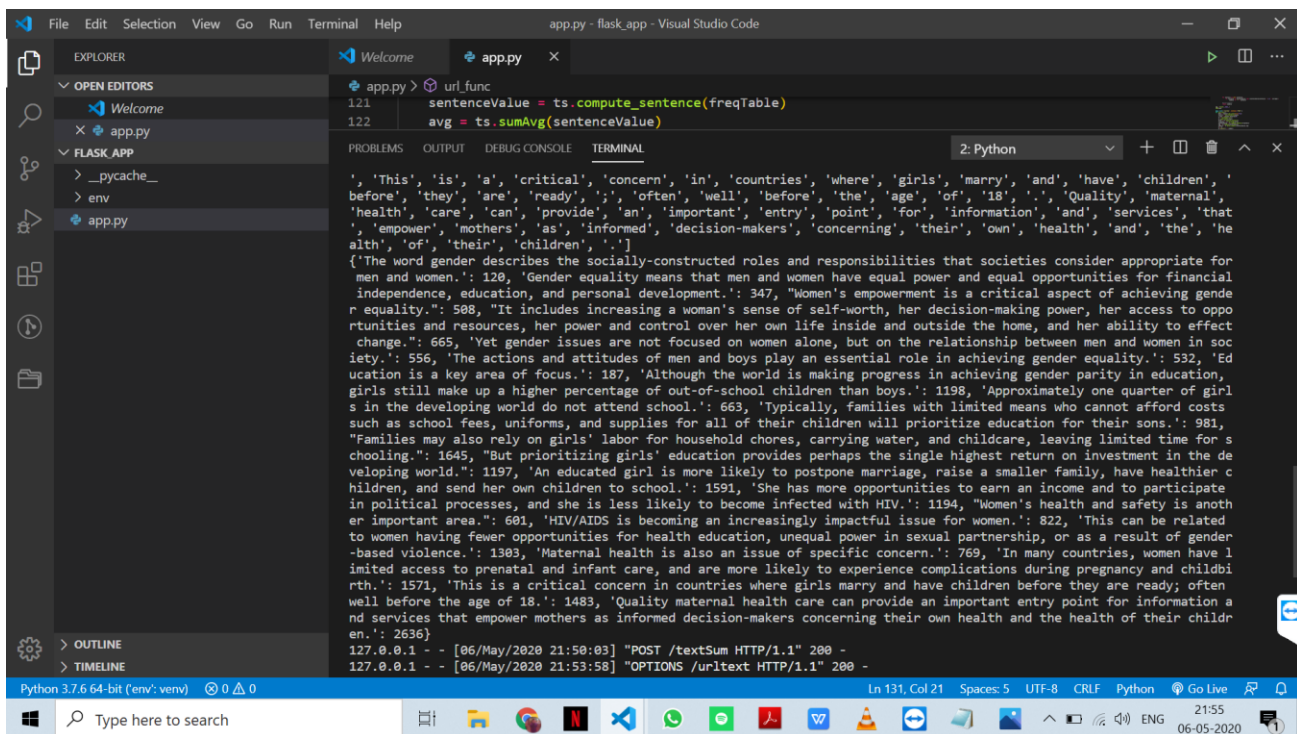
Summarized Output



Output Explained



Output Explained



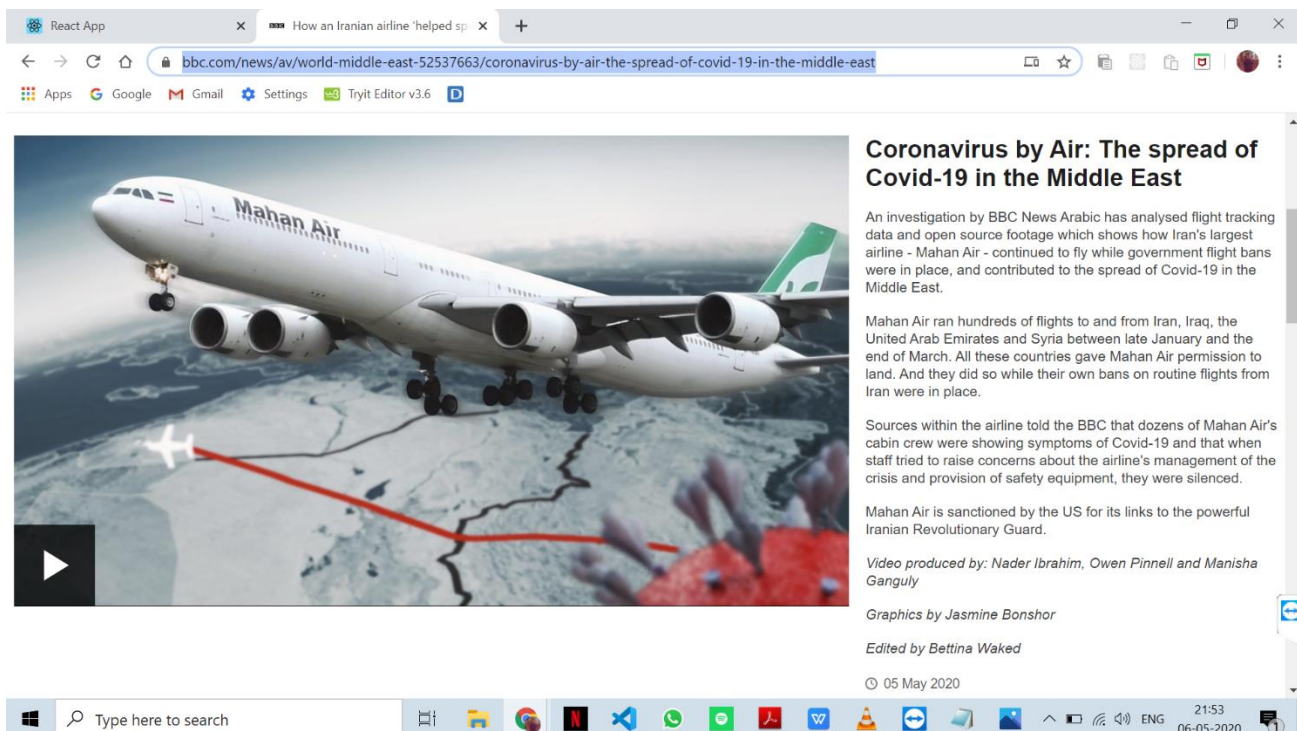
The screenshot shows a Visual Studio Code editor window with a file named `app.py` open. The code in the editor is as follows:

```
121 sentenceValue = ts.compute_sentence(freqTable)
122 avg = ts.sumAvg(sentenceValue)
```

The output window at the bottom displays the following text:

```
Python 3.7.6 64-bit (env: venv) 0 0 0
Ln 131, Col 21 Spaces: 5 UTF-8 CRLF Python Go Live
127.0.0.1 - - [06/May/2020 21:50:03] "POST /textSum HTTP/1.1" 200 -
127.0.0.1 - - [06/May/2020 21:53:58] "OPTIONS /urltext HTTP/1.1" 200 -
```

URL that needs to be summarized



The screenshot shows a web browser window with the URL `bbc.com/news/av/world-middle-east-52537663/coronavirus-by-air-the-spread-of-covid-19-in-the-middle-east`. The article is titled "Coronavirus by Air: The spread of Covid-19 in the Middle East". The main image shows a Mahan Air airplane flying over a map of the Middle East, with a red line indicating a flight path. The article text is as follows:

Coronavirus by Air: The spread of Covid-19 in the Middle East

An investigation by BBC News Arabic has analysed flight tracking data and open source footage which shows how Iran's largest airline - Mahan Air - continued to fly while government flight bans were in place, and contributed to the spread of Covid-19 in the Middle East.

Mahan Air ran hundreds of flights to and from Iran, Iraq, the United Arab Emirates and Syria between late January and the end of March. All these countries gave Mahan Air permission to land. And they did so while their own bans on routine flights from Iran were in place.

Sources within the airline told the BBC that dozens of Mahan Air's cabin crew were showing symptoms of Covid-19 and that when staff tried to raise concerns about the airline's management of the crisis and provision of safety equipment, they were silenced.

Mahan Air is sanctioned by the US for its links to the powerful Iranian Revolutionary Guard.

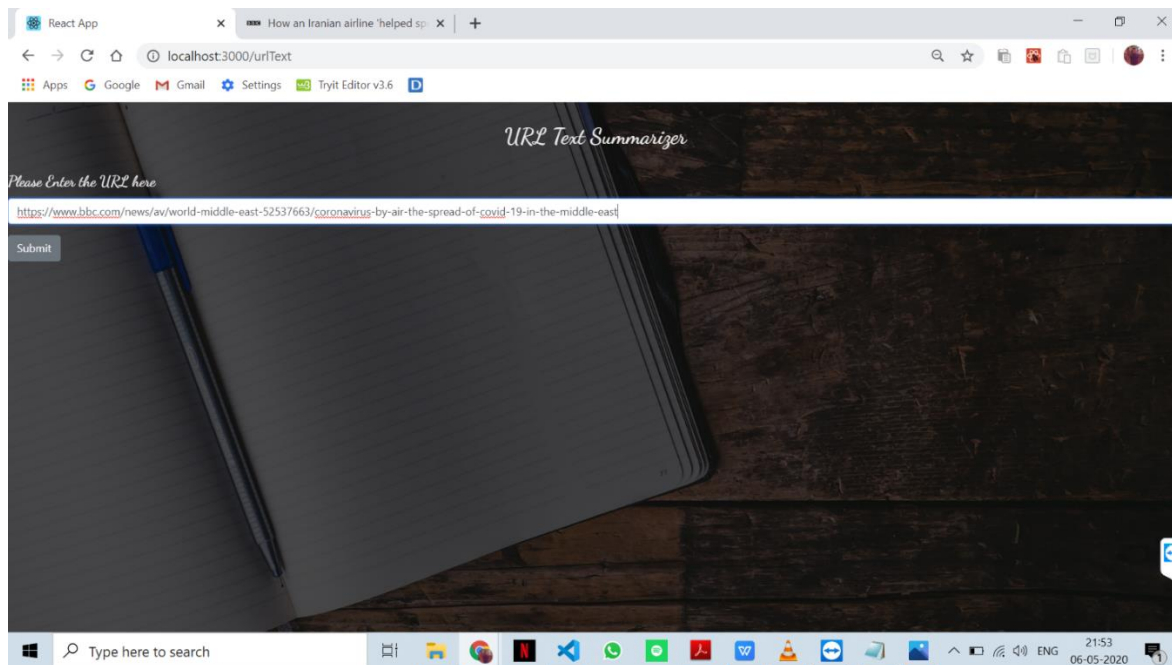
Video produced by: Nader Ibrahim, Owen Pinnell and Manisha Ganguly

Graphics by Jasmine Bonshor

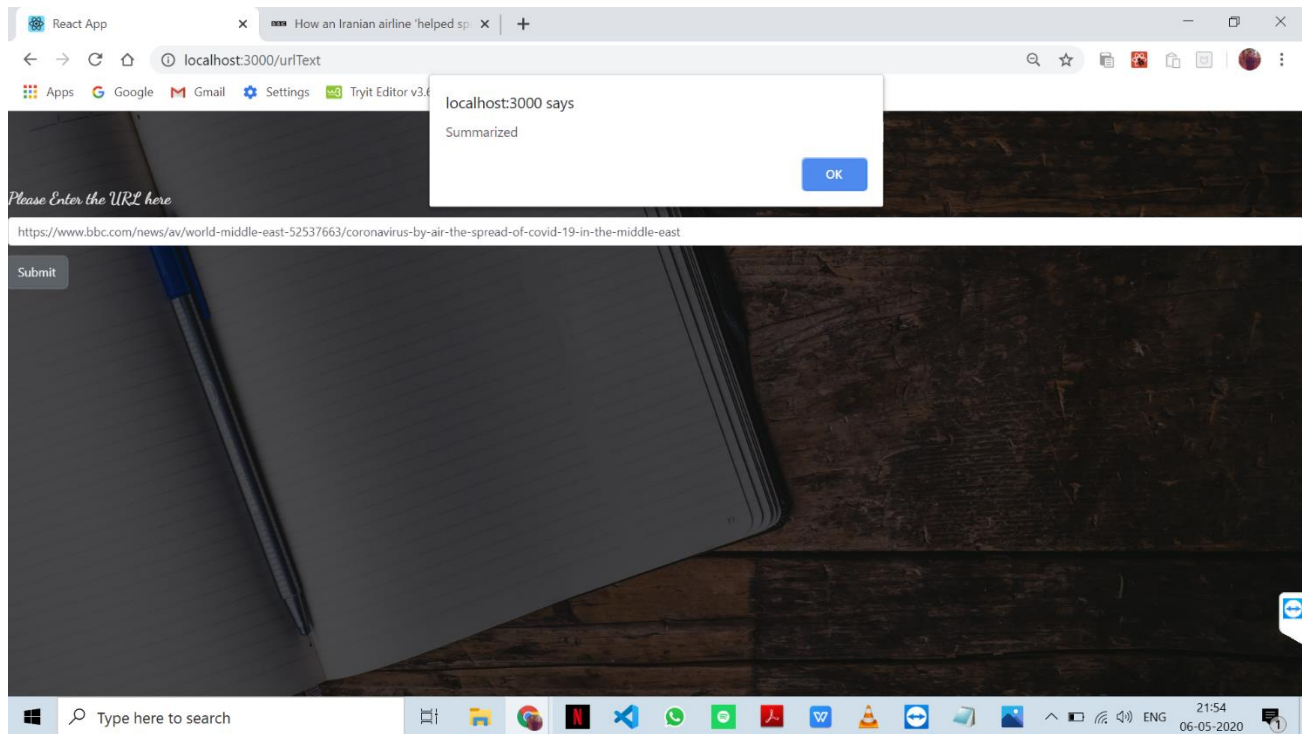
Edited by Bettina Waked

05 May 2020

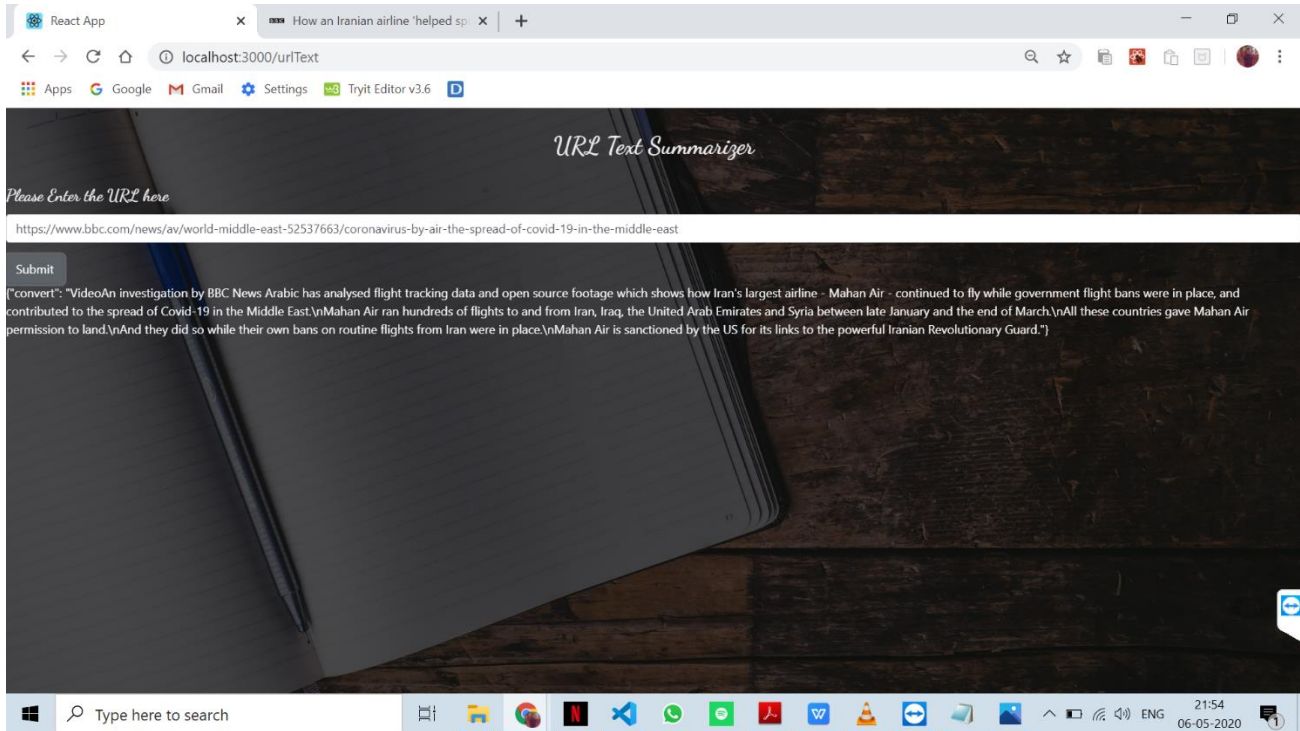
URL Text Summarizer



URL text summarized



Text Summarized From The Given URL



CHAPTER 6

CONCLUSION

As with time internet is growing at a very fast rate and with it data and information is also increasing. it will going to be difficult for human to summarize large amount of data. Thus there is a need of automatic text summarization because of this huge amount of data.

There are multiple automatic text summarizers with great capabilities and giving good results. We have learned all the basics of Extractive and Abstractive Method of automatic text summarization and tried to implement extractive one.

CHAPTER 7

REFERENCES

- ❖ Python Text Processing with NLTK by Perkins
- ❖ An Introduction to Natural Language Processing by Jufrasky, Martin
- ❖ Text Summarizer by Dan Jurafsky and Chris Manning
- ❖ "Survey of the State of the Art in Human Language Technology" (CUP, ed. Cole., R., 1997) by K. Sparck Jones called "Summarization".