A
Minor Project-I Report
On

"TEXT SUMMARIZATION"

Submitted in partial fulfillment of
The requirements for the 6th Semester Sessional Examination
of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE & ENGINEERING

By
Neeraj Kumar(20UG010186)

Under the Supervision of

Mrs. Rasmita Panigrahi

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GIET UNIVERSITY, GUNUPUR

2023 - 24

GIET UNIVERSITY,GUNUPUR

Dist. - Rayagada, Odisha-765022, Contact:- +91 7735745535, 06857-250170,172, Visit us:- **www.giet.edu**

Department of Computer Science & Engineering

CERTIFICATE

This is to certify that the project work entitled "TEXT SUMMARIZATION " is done by Name-Neeraj Kumar in partial fulfillment of the requirements for the 6th Semester Sessional Examination of Bachelor of Technology and Computer Science and Engineering during the academic year 2023-24. This work is submitted to the department as a part of evaluation of 6th Semester Major Project-I.

Project Supervisor *Class Teacher*

Project Coordinator, 4th Year ***HoD, CSE****, 4th Year*

ACKNOWLEDGEMENT

I would like to thank my supervisor Mrs.Rasmita Panigrahi for placing complete faith and confidence in our ability to carry out this project and advice at the crucial junctures and for showing me the right way. Secondly I would like to thank my class teacher Mr.Ranjit panigrahi for providing us with the right guidance. Thirdly I would like to thanks our Project Coordinator Mr.Bhavani Sankar Panda for being the coordinator of the project. A special gratitude to our HOD Dr.Bandita Sahu, for allowing us to use the facilities available.

I would like to thank the other faculty members also, at this occasion. Last but not the least, I would like to thank my friends and family for the support and encouragement they have given me during the course of our work.

Name of Students : -

Neeraj Kumar

ABSTRACT

The past decade has endorsed a great rise in Artificial Intelligence. Text summarization which comes under AI has been an important research area that identifies the relevant sentences from a piece of text. By Text Summarization, we can get short and precise information by preserving the contents of the text. This project presents an approach for generating a short and precise extractive summary for the given document of text. A cumulative method using statistical methods and neural network model for extractive text summarization of sports articles using extraction of various features is discussed in this project. The features taken are TF-ISF, Sentence Length, Sentence Position, Sentence to Sentence cohesion, Proper noun, Pronoun. Each sentence is given a score known as the predictive score is calculated and the summary for the given document of text is given based on the predictive score or also known as the rank of the sentence. The accuracy is checked using the DUC dataset and sports articles of various newspapers like the New York Times, CNN. The precision of 73% is acquired when compared with System Generated Summary (SGS) and manual summary, on an average.

Keywords- Artificial Intelligence, Natural Language Processing, Term frequency inverse sentence frequency, cosine similarity, System Generated Summary (SGS).

INDEX

INTRODUCTION

TEXT MINING

Text mining can also be referred as Text analysis, which uses different Artificial Intelligence technologies. Text mining can be referred to as deriving high-quality information by drawing patterns and identifying the important keywords from the unstructured data. Text mining tasks include text classification which is the classification of the text for example genre, the sentimental analysis which tells about how the author wrote the sentences that is in which tone, document clustering refers to as clustering the documents using unsupervised learning, text summarization for obtaining a short and precise text. Each task is different from each other and has its own probe and methodologies. The purpose is too unstructured information, extract meaningful numeric indices from the text. Thus, make the information contained in the text accessible to the various algorithms. Information can be extracted to derive summaries contained in the documents. Hence, you can analyze words, clusters of words used in documents. In the most general terms, text mining will "turn text into numbers". Such as predictive data mining projects, the application of unsupervised learning methods.

Information Extraction

This is the most famous text mining technique. Information exchange refers to the process of extracting meaningful information from vast chunks of textual data. This text mining technique focuses on identifying the extraction of entities, attributes, and their relationships from semi-structured or unstructured texts. Whatever information is extracted is then stored in a database for future access and retrieval. The efficacy and relevancy of the outcomes are checked and evaluated using precision and recall processes. In most of the cases, this activity includes processing human language texts by means of NLP.

Information Retrieval Information Retrieval

(IR) refers to the process of extracting relevant and associated patterns based on a specific set of words or phrases. In this text mining technique, IR systems make use of different algorithms to track and monitor user behaviors and discover relevant data accordingly. Information retrieval is regarded as an extension to document retrieval. That the documents that are returned are processed to condense. Thus, document retrieval follows by a text summarization stage. That focuses on the query posed by the user. IR systems help in to narrow down the set of documents that are relevant to a problem. As text mining involves applying very complex algorithms to large document collections. Also, IR can speed up the analysis significantly by reducing the number of documents. Google and Yahoo search engines are the two most renowned IR systems.

Categorization

This is one of those text mining techniques that is a form of "supervised" learning wherein normal language texts are assigned to a predefined set of topics depending upon their content. Categorization in text mining means sorting documents into groups. Automatic document classification uses a combination of natural language processing (NLP) and machine learning to categorize customer reviews, support tickets, or any other type of text document based on their contents. Thus, categorization or rather Natural Language Processing (NLP) is a process of gathering text documents and processing and analyzing them to uncover the right topics or indexes for each document. The co-referencing method is commonly used as a part of NLP to extract relevant synonyms and abbreviations from textual data.

Clustering

Clustering is one of the most crucial text mining techniques. It seeks to identify intrinsic structures in textual information and organize them into relevant subgroups or 'clusters' for further analysis. A significant challenge in the clustering process is to form meaningful clusters from the unlabelled textual data without having any prior information on them.

Cluster analysis is a standard text mining tool that assists in data distribution or acts as a pre-processing step for other text mining algorithms running on detected clusters.

Summarization

Text summarization refers to the process of automatically generating a compressed version of a specific text that holds valuable information for the end-user. The aim of this text mining technique is to browse through multiple text sources to craft summaries of texts containing a considerable proportion of information in a concise format, keeping the overall meaning and intent of the original documents essentially the same. Text summarization integrates and combines the various methods that employ text categorization like decision trees, neural networks, regression models, and swarm intelligence. There are broadly two different approaches that are used for text summarization: Extractive text summarization: The name itself gives what this approach does. We identify the important sentences or phrases from the original text and extract only those from the text. Those extracted sentences would be our summary. Abstractive summarization: 4 This is a very interesting approach. Here, we generate new sentences from the original text. This contrasts with the extractive approach we saw earlier where we used only the sentences that were present. The sentences generated through abstractive text summarization might not be present in the original text.

NATURAL LANGUAGE PROCESSING

Natural Language Processing (NLP) is all about computer and human languages interactions. A finest procedure that helps the system in reading and understanding the given text. Natural Language Processing, or NLP for short, is broadly defined as the automatic manipulation of natural language, like speech and text, by software. Data generated from conversations, declarations or even tweets are examples of unstructured data. Unstructured data doesn’t fit neatly into the traditional row and column structure of relational databases and represent most data available in the actual world. It is messy and hard to manipulate.

Text Embedding

Text Embeddings are real valued vector representations of strings. We build a dense vector for each word, chosen so that it's similar to vectors of words that appear in similar contexts. Word embeddings are considered a great starting point for most deep NLP tasks. They allow deep learning to be effective on smaller datasets, as they are often the first inputs to a deep learning architecture and the most popular way of transfer learning in NLP. The most popular names in word embeddings are Word2vec by Google (Mikolov) and GloVe by Stanford (Pennington, Socher and Manning). Let's delve deeper into these word representations.

Neural Machine Translation

Neural Machine Translation is the approach of modelling this entire process via one big artificial neural network, known as a Recurrent Neural Network (RNN). RNN is a stateful neural network, in which it has connections between passes, connections through time. Neurons are fed information not just from the previous layer but also from themselves from the previous pass. This means that the order in which we feed the input and train the network matters: feeding it "Donald" and then "Trump" may yield different results compared to feeding it "Trump" and then "Donald". Neural Machine Translation is a machine translation approach that applies a large artificial neural network toward predicting the likelihood of a sequence of words, often in the form of whole sentences. Unlike statistical machine translation, which consumes more memory and time, neural machine translation, NMT, trains its parts end-to-end to maximize performance. NMT systems are quickly moving to the forefront of machine translation, recently outcompeting traditional forms of translation systems.

DEEP LEARNING

Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks. Deep learning is a subset of machine learning in artificial intelligence (AI) that has networks capable of learning unsupervised

from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network. It has been evolved from Biological neural network.

Biological neural network

Computational modelling and theoretical analysis of biological neural networks are integral parts of computational neuroscience. This field's association with cognitive and behavioral modelling is derived from the fact that biological neural systems maintain very close relationships to this modelling. The hope is that through this modelling a link can be established between the data obtained from the biological processes, the probable working mechanism of the biological network, and learning through statistics and theory of information. The neural syst

PURPOSE

Text summarization has become one of the hottest topics in Text mining as there is more amount of text that is produced on the internet. So, has become so difficult to read all the content due to the busy schedules and as the time has become the more important constraint in everyone's life. To make the text content simpler to read, there came an idea of summarizing 26 the text by giving only the important information from the available text articles. The sports are the most admiring topics in news, and everyone wants to know about the sports news and due to the time constraint, they couldn't read the whole news which contains only a few important information. So, we have chosen to summarize the sports articles.

PROJECT SCOPE

Generating a summary from a text document. So that it helps in understanding a huge text data. With the availability of internet at every corner of the world, the amount of information is growing rapidly at certain rate. Considering the hectic schedule of the people and the need for information abstraction, here comes the summarization of the News Articles. Every domain in the news has its own type of information important. So, as it is difficult to summarize all types of news data. We are limiting our problem to extractive summarize the sports genre in news articles.

SYSTEM ANALYSIS

HARDWARE REQUIRMENTS:~

- Operating System: Windows 11
- Processor: Intel core i5
- Ram: 8GB
- Hard Disk: 1TB

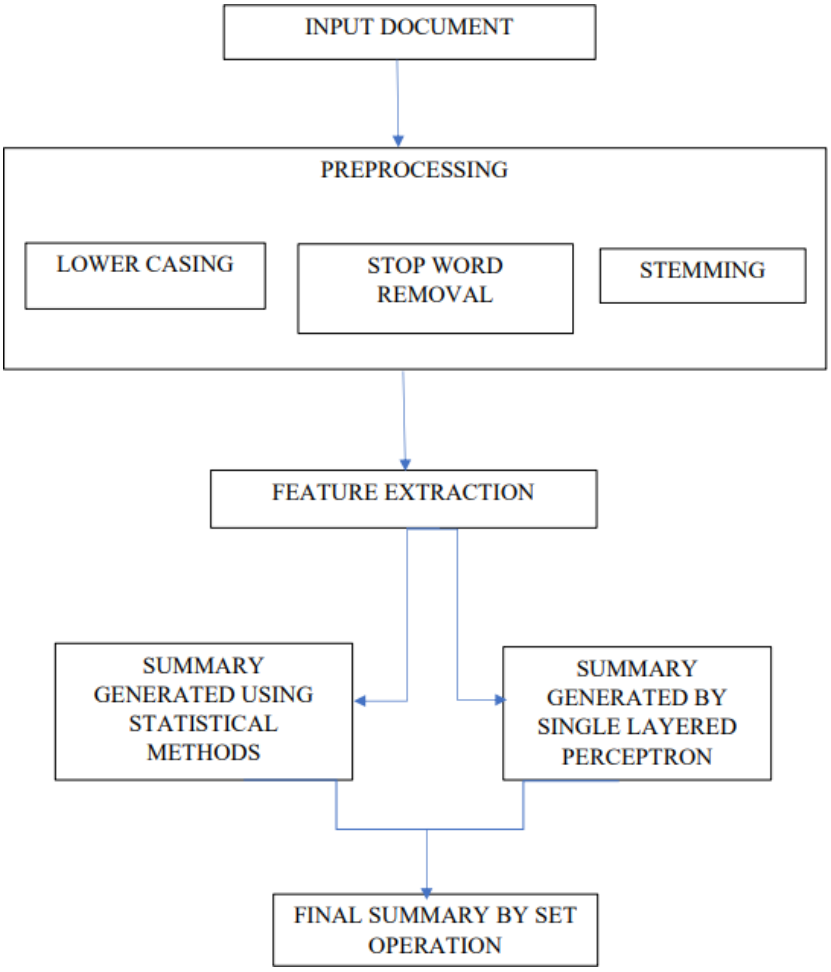
SOFTWARE REQUIRMENTS:~

- Languages: Python, Flask.
- Framework: PyCharm, Natural language toolkit
- Development Platform: Google colab

SYSTEM DESIGN & SPECIFICATION

PROPOSED SYSTEM

- **ARCHITECTURE**



- **Detailed Understanding of architecture:**

Input Document:

To generate a summary some input should be given in the file format ".txt". The text documents of sports articles are considered for summarization and are valid since the author has proposed a method for summarizing sports articles. The input document should only be in English language. The size of the input document can be large as well.

Pre-processing:

The input text is divided into sentences based on the sentence terminator.
These sentences are individually preprocessed using the below three techniques:

a. Lower casing: In the lower casing, the whole document is converted into lowercase alphabetical letters. This is done to remove the ambiguity of the words with different casing. This is done using the lower() of python language which is available inbuilt. There is another method also, considering each character and its ascii values. Add the required value to convert to lower case.
b. Stop word removal: The stop words in the document are removed from the document because these are the most frequent words like articles, prepositions, conjunctions which don't add any value in defining the importance of a sentence. The stop words are removed using nltk (natural language tool kit) library. All the stop words are stored in a list and each word is compared with the words present in the list. If the word is present in the list it is removed from the document, else it is remained unaltered.
c. Stemming: Stemming is referred as getting the words reduced to their root form. It removes the different forms of the same word present in different sentences. In this porters stemmer is used.

3) Feature Extraction:

a. TF-ISF: Term frequency and inverse document frequency (TF-IDF) are used for information retrieval systems. In this paper, the summarization is done for a single document of sports article and Term frequency inverse sentence frequency (TF-ISF) is calculated.
b. Sentence length: This feature provides the less weightage to short sentences as short sentences are relatively less important when compared to long sentences. It is measured by calculating the ratio of no of words in the sentence to the no of words in the longest sentence of the document.
c. Sentence position: In a document the numerical value of the sentence's position is gauged (4). The position is evaluated as the normalized percentile score in the range of 0 to 1.
d. Cosine similarity: Analogy between sentences is calculated using this property. This briefs how much similar this sentence with other sentences in a document is. A vector for each sentence is assessed and (5) is practiced obtaining the score of the feature.
e. Existence of proper noun: Proper names broaching to places and people might be useful to decide the relevance of a sentence. This binary feature, with value '1' if a sentence contains these proper names and 0 otherwise.
f. Existence of pronoun: Pronouns cites to the persons which are described in the previous sentences. This binary feature, with value '1' if a sentence contains these proper names and 0 otherwise. For Example: He, she, it, they.
g. Sentence containing scores: Since in sports the score of the team or the individual participant is important the score is taken into consideration. This binary feature, with value '1' if a sentence contains these proper names and 0 otherwise.

4) Summary generated using Statistical methods:

The mean, median and mode are generally some statistical tools used. The mean is used in many cases. Firstly, the input document is divided in to sentences and then the sentences are divided into words. Secondly, the preprocessing which consists of lower casing, stop word removal and stemming is done and stored in a list.

5) Summary generated using Single layered perceptron:

The single layered perceptron is used as a neural network. It consists of one input layer, one hidden layer and an output layer. Input layer has 7 nodes and output layer has one node. The values of the features become the input to the neural network which is multiplied by weights and it is passed through activation function, sigmoid is chosen and we get a value for each sentence and it becomes the predictive score of a sentence.

Final summary by set operation:

The common sentences are taken from both the techniques and the summary is given based on the compression ratio considering

RESULT & ANALYSIS

The evaluation measures used are for context evaluation which comes under co-selection. In co-selection there are three types of measures they are precision, recall, f-measure. Precision (P) can be defined as the fraction of the number of sentences common in manual and automated summary to the number of sentences in the automated summary. Precision (P) can be procured using equation (6).

Recall (R) can be defined as the ratio of the number of sentences common in manual and automated summary to the manual summary. The recall can be calculated using equation (7).

F-measure (F) can be defined as the harmonic mean of precision and recall. F-measure (F) can be procured using equation (8).

Dataset:

The dataset used is the BBC sports corpus since it is the worldwide famous for its news. It consists of 737 articles for five different types of sports namely athletics, cricket, football, rugby, tennis and 101, 124, 265, 147, 100 articles respectively. The model is tested for these articles and performed with a good amount of accuracy by generating the relevant context leaving the unnecessary sentences.

The System generated summary (SGS) is tested for accuracy with a manual summary and an online summarizer tool. The manual summary is given by one of the literates who have secured a degree 65 in Literature. The online tool used for generating the summary is

Title	Manual w.r.t SGS			Manual w.r.t online summarizer tool		
	P	R	F	P	R	F
Doc-1	0.8	0.8	0.8	0.667	0.8	0.7274
Doc-2	0.7142	0.625	0.6662	0.6667	0.5	0.5714
Doc-3	0.7142	0.8333	0.7689	0.5	0.667	0.5714
Doc-4	0.75	0.75	0.75	0.667	0.5	0.5714
Doc-5	0.6842	0.7647	0.7219	0.7334	0.6470	0.6874
Doc-6	0.7142	0.7142	0.7142	0.5714	0.5714	0.5714
Doc-7	0.7	0.7368	0.7245	0.834	0.834	0.834
Doc-8	0.7142	0.7692	0.7401	0.5834	0.5384	0.56
Doc-9	0.8	0.889	0.8336	0.667	0.667	0.667
Doc-10	0.75	0.8	0.7741	0.5625	0.6	0.58

Table 1: Comparison of performance between manual w.r.t SGS and manual w.r.t online summarizer tool Text Compactor. The author has considered the compression ratio half of the original document.

Firstly, a manual summary is compared with SGS and a manual summary is compared with online summarizer tool for precision, recall, and f-measure. The documents tested are the sports articles from various newspapers including the New York Times, CNN.

For analyzing the accuracy, the author also checked it with DUC-2002. The Table-1 shows the number of sentences in each document taken from the news articles. The data of 10 documents is shown.

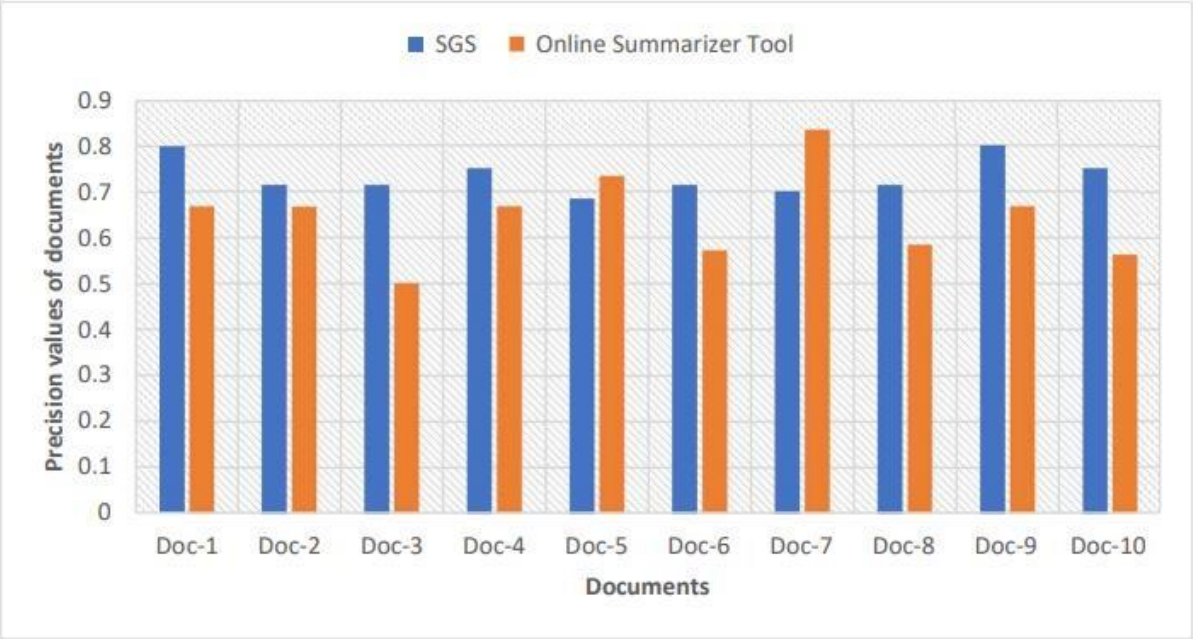
Title	Number of sentences in document	Number of sentences in SGS (α=50%)
Doc-1	11	5

Doc-2	16	7
Doc-3	15	7
Doc-4	8	4
Doc-5	36	19
Doc-6	15	7
Doc-7	39	20
Doc-8	27	14
Doc-9	19	10
Doc-10	32	16

Table-2: Shows number of sentences in document as well as SGS.

	Average		
	P	R	F
Manual w.r.t SGS	0.7341	0.7681	0.7493
Manual w.r.t online summarizer tool	0.6452	0.6328	0.6341

Table-3: The average of the Precision(P), recall(R), f-measure(F) for all the 10 documents



Comparison between SGS and Online summarizer tool

CODING

```
tokens = []
for token in nltk.word_tokenize(formatted_text):
    tokens.append(token)
tokens = [word for word in tokens if word not in stopwords and word not in string.punctuation]
formatted_text = ' '.join(element for element in tokens)
word_frequency = nltk.FreqDist(nltk.word_tokenize(formatted_text))
highest_frequency = max(word_frequency.values())
for word in word_frequency.keys():
    # print(word)
    word_frequency[word] = (word_frequency[word] / highest_frequency)
sentence_list = nltk.sent_tokenize(text)
score_sentences = {}
for sentence in sentence_list:
    # print(sentence)
    for word in nltk.word_tokenize(sentence.lower()):
        # print(word)
        if sentence not in score_sentences.keys():
            score_sentences[sentence] = word_frequency[word]
        else:
            score_sentences[sentence] += word_frequency[word]
best_sentences = heapq.nlargest(3, score_sentences, key=score_sentences.get)
summary = ' '.join(best_sentences)
#for keyword extraction
nlp = spacy.load("en_core_web_sm")
# add PyTextRank to the spaCy pipeline
nlp.add_pipe("textrank")
doc = nlp(text)
keyword=[]
for phrase in doc._.phrases[:10]:
    keyword.append(phrase.text)
```

```
text = ' '.join(map(lambda p: p.text, soup.find_all('p')))
original_text = soup.title.text, text
original_text = str(original_text)
TAG_RE = re.compile(r'<[^>]+>')
text=TAG_RE.sub('', original_text)
nltk.download('punkt')
nltk.download('stopwords')
stopwords = nltk.corpus.stopwords.words('english')
string.punctuation
formatted_text = text.lower()
tokens = []
for token in nltk.word_tokenize(formatted_text):
    tokens.append(token)
tokens = [word for word in tokens if word not in stopwords and word not in string.punctuation]
formatted_text = ' '.join(element for element in tokens)
word_frequency = nltk.FreqDist(nltk.word_tokenize(formatted_text))
highest_frequency = max(word_frequency.values())
for word in word_frequency.keys():
    # print(word)
    word_frequency[word] = (word_frequency[word] / highest_frequency)
sentence_list = nltk.sent_tokenize(text)
score_sentences = {}
for sentence in sentence_list:
    # print(sentence)
    for word in nltk.word_tokenize(sentence.lower()):
        # print(word)
        if sentence not in score_sentences.keys():
            score_sentences[sentence] = word_frequency[word]
        else:
            score_sentences[sentence] += word_frequency[word]
```

```
language = request.form.get('language')
if(language=="English"):
    return render_template("predict.html", summary=summary, keyword=keyword)
elif(language=="Hindi"):
    translator = Translator()
    translation = translator.translate(summary, dest='hi')
    summary = translation.text
    return render_template("predict.html", summary=summary, keyword=keyword)
elif(language=="Odia"):
    translator = Translator()
    translation = translator.translate(summary, dest='or')
    summary = translation.text
    return render_template("predict.html", summary=summary, keyword=keyword)
elif (language == "Telugu"):
    translator = Translator()
    translation = translator.translate(summary, dest='te')
    summary = translation.text
    return render_template("predict.html", summary=summary, keyword=keyword)
elif (language == "Tamil"):
    translator = Translator()
    translation = translator.translate(summary, dest='ta')
    summary = translation.text
    return render_template("predict.html", summary=summary, keyword=keyword)
pp.route('/summarize', methods=['POST'])
f summarize():
url=request.form.get('url')
req = urllib.request.Request(url)
page = urllib.request.urlopen(req).read().decode('utf8')
soup = BeautifulSoup(page, "html.parser")
```

```
  elif (language == "Tamil"):
     translator = Translator()
     translation = translator.translate(summary, dest='ta')
     summary = translation.text
     return render_template("predict.html", summary=summary, keyword=keyword)

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

```
from flask import Flask, render_template,request
app = Flask(__name__,template_folder='template')
import re
import urllib.request
from bs4 import BeautifulSoup
import nltk
import string
import heapq
import spacy
from googletrans import Translator
@app.route("/")
def index():
    return render_template("index.html")
@app.route("/template/input")
def inpt():
    return render_template("input.html")
@app.route("/template/link")
def lnk():
    return render_template("link.html")
@app.route("/template/profile")
def pro():
    return render_template("profile.html")
@app.route('/predict', methods=['POST'])
def predict():
    text=request.form.get('text')
    nltk.download('punkt')
    nltk.download('stopwords')
    stopwords = nltk.corpus.stopwords.words('english')
    string.punctuation
    formatted_text = text.lower()
```

```
best_sentences = heapq.nlargest(15, score_sentences, key=score_sentences.get)
summary = ' '.join(best_sentences)

nlp = spacy.load("en_core_web_sm")
# add PyTextRank to the spaCy pipeline
nlp.add_pipe("textrank")
doc = nlp(formatted_text)
keyword = []
for phrase in doc._.phrases[:10]:
    keyword.append(phrase.text)

language = request.form.get('language')
if (language == "English"):
    return render_template("predict.html", summary=summary, keyword=keyword)
elif (language == "Hindi"):
    translator = Translator()
    translation = translator.translate(summary, dest='hi')
    summary = translation.text
    return render_template("predict.html", summary=summary, keyword=keyword)
elif (language == "Odia"):
    translator = Translator()
    translation = translator.translate(summary, dest='or')
    summary = translation.text
    return render_template("predict.html", summary=summary, keyword=keyword)
elif (language == "Telugu"):
    translator = Translator()
    translation = translator.translate(summary, dest='te')
    summary = translation.text
    return render_template("predict.html", summary=summary, keyword=keyword)
elif (language == "Tamil"):
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
   <form action="/summarize" method="POST">
        <input type="url" name="url" id="url" placeholder="Enter the link here"><br>
        <label for="language">Choose a language:</label>
        <select name="language" id="language">
            <option value="English">English</option>
            <option value="Hindi">Hindi</option>
            <option value="Odia">Odia</option>
            <option value="Telugu">Telugu</option>
            <option value="Tamil">Tamil</option>
        </select><br>
        <input type="submit"  value="Summarize">
   </form>
</body>
</html>
```

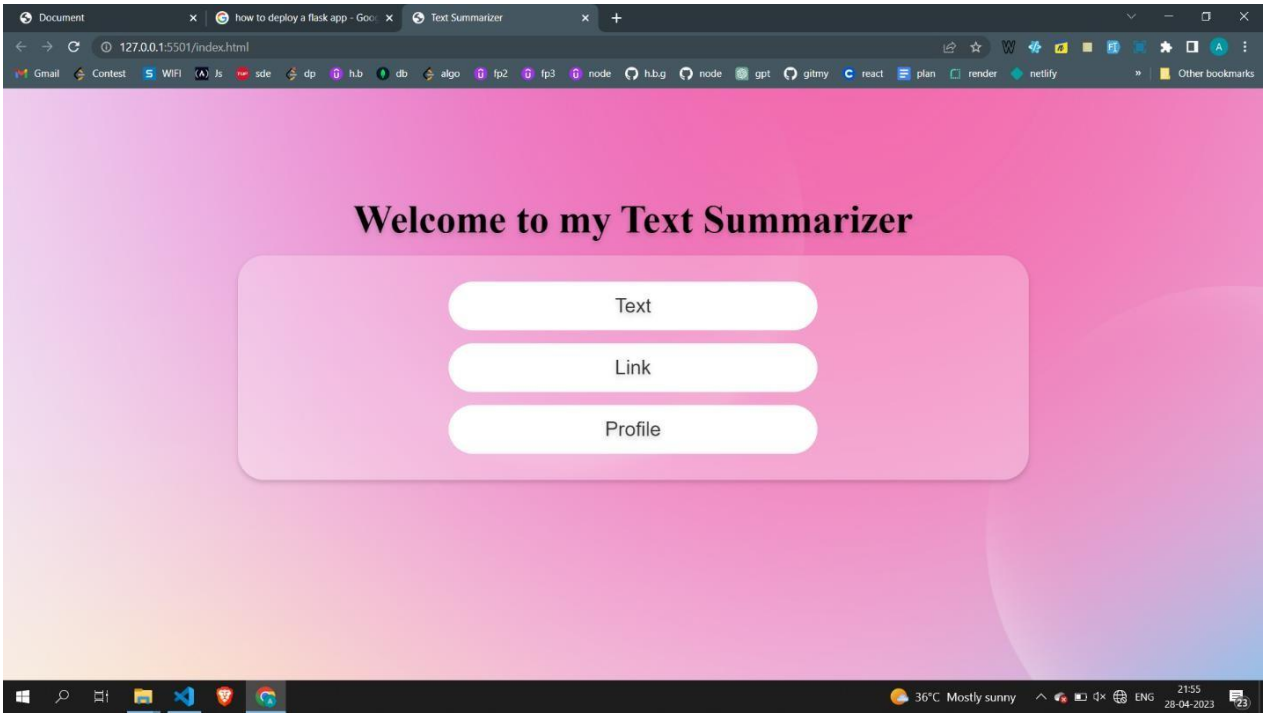
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>profile page</h1>
</html>
```

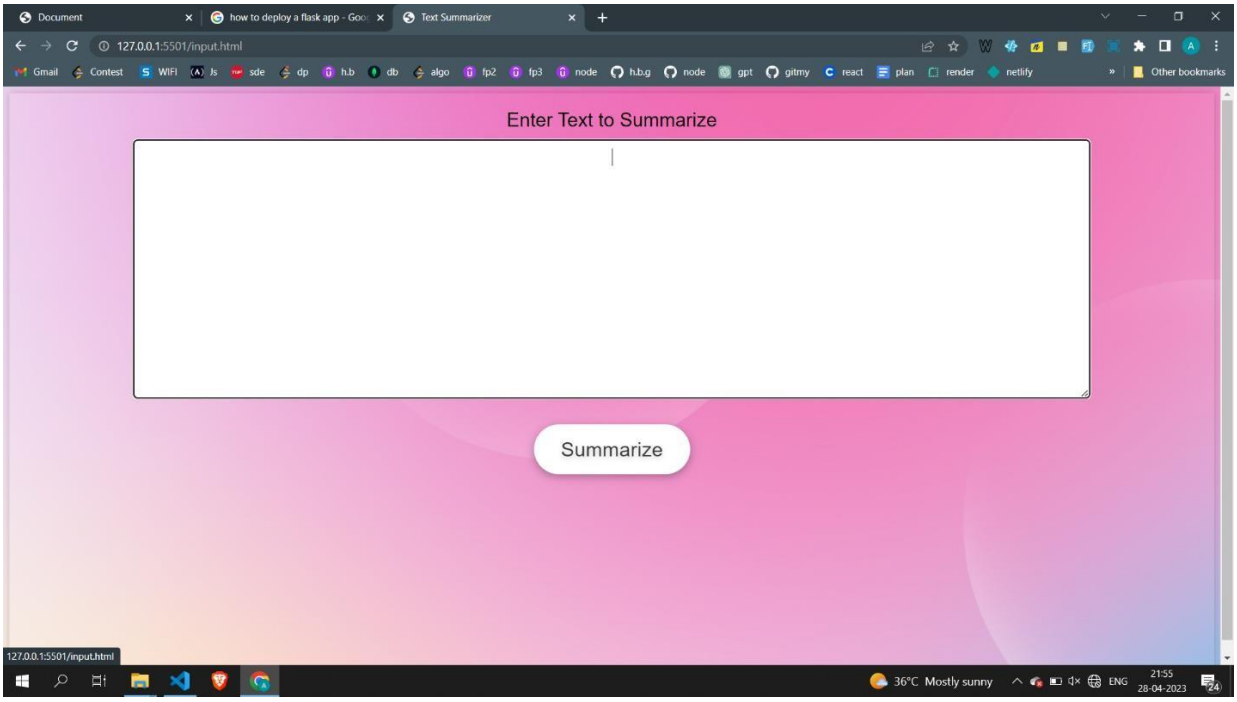
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
  <h1>OUTPUT DATA</h1>
  <p>{{summary}}</p>
  <h1>Keywords</h1>


  <p>{{keyword[0]}}</p>
  <p>{{keyword[1]}}</p>
  <p>{{keyword[2]}}</p>
  <p>{{keyword[3]}}</p>
  <p>{{keyword[4]}}</p>
  <p>{{keyword[5]}}</p>
</body>
```

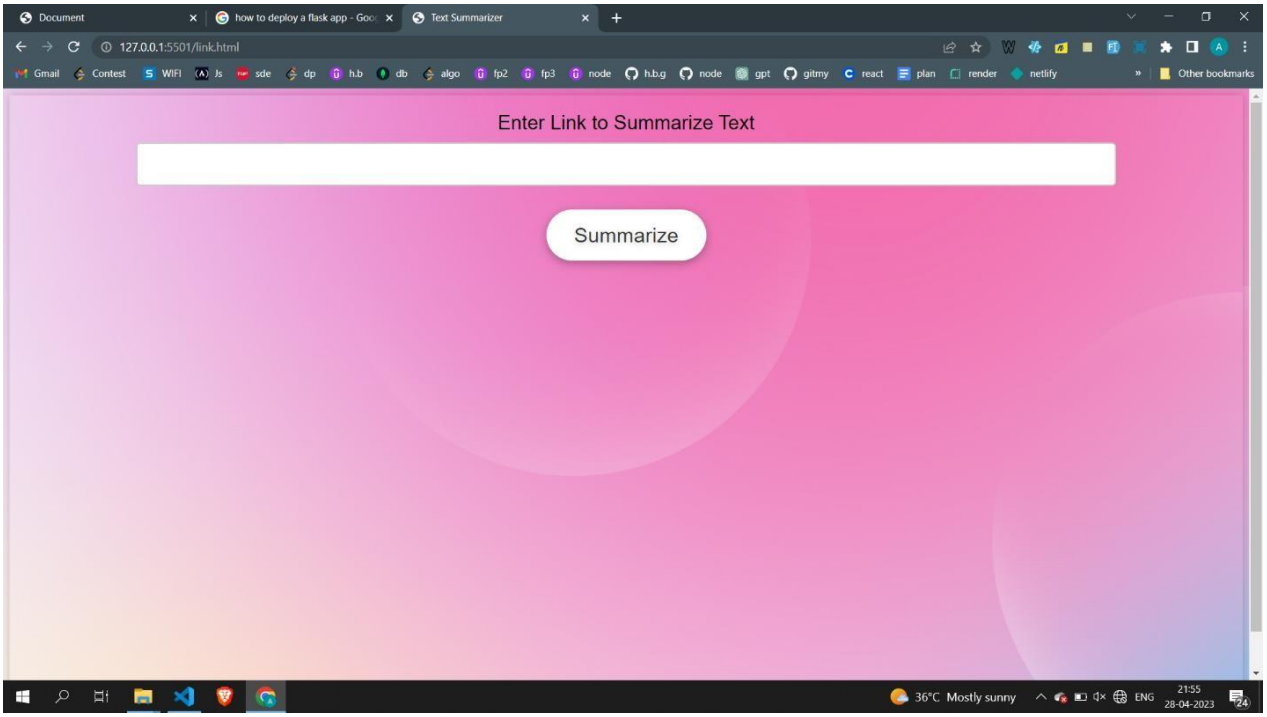
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <button>  <a href="template/input">input</a> </button>
    <button><a href="template/link">Link</a></button>
    <button><a href="template/profile">Profile</a></button>
</body>
</html>
```

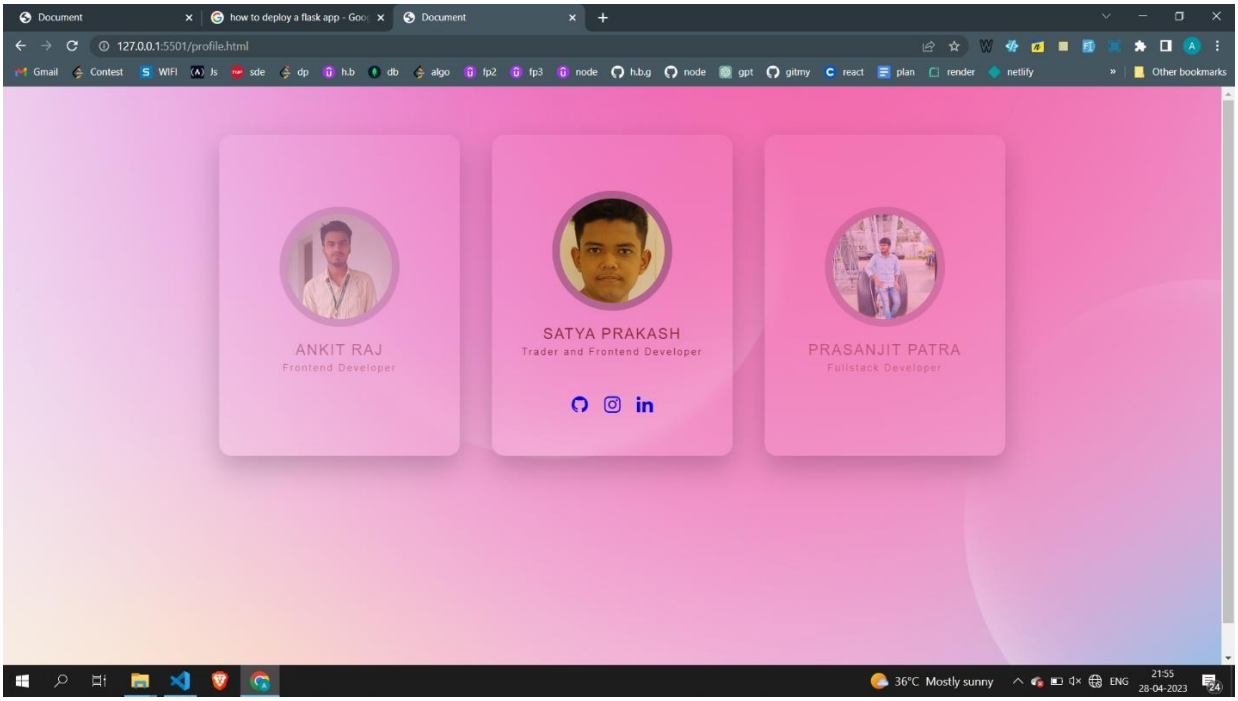
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <button>  <a href="template/input">input</a> </button>
    <button><a href="template/link">Link</a></button>
    <button><a href="template/profile">Profile</a></button>
</body>
</html>
```

RESULT


Document
how to deploy a flask app - Goo
Text Summarizer
127.0.0.1:5501/index.html
Welcome to my Text Summarizer
Text
Link
Profile
36°C Mostly sunny
ENG
21:55
28-04-2023


Document
how to deploy a flask app - Goo
Text Summarizer
127.0.0.1:5501/input.html
Enter Text to Summarize
Summarize
127.0.0.1:5501/input.html
36°C Mostly sunny
ENG
21:55
28-04-2023


Document
how to deploy a flask app - Goo
Text Summarizer
127.0.0.1:5501/link.html
Enter Link to Summarize Text
Summarize
36°C Mostly sunny
ENG
21:55
28-04-2023


Document
how to deploy a flask app - Goo
Document
127.0.0.1:5501/profile.html
ANKIT RAJ
Frontend Developer
SATYA PRAKASH
Trader and Frontend Developer
PRASANJIT PATRA
Fullstack Developer
36°C Mostly sunny
ENG
21:55
28-04-2023

CONCLUSION

- The extractive text summarization has many tasks involved in generating the précis. The author proposed a statistical approach for generating an extractive précis of a sports articles that uses sentence ranking based on the selected features for the sentences. The most critical part is identifying the important sentences without loss of meaning from the given document. Using the proposed method on an average, 73% precision, 76% recall, and 74% f-measure is obtained which when compared with a manual generated summary and an online summarizer tool. the future, we may investigate deep learning algorithms to see the results.

Future work:

- In nearly future, the proposed model can also be developed to a neural network model that takes the values of different features as input for generating an accurate precis by considering the contextual meaning. To increase the accuracy of the precis, various features can also be added.