TEXT SUMMARIZATION

Vellore Institute of Technology, Vellore

Team Members

Devavrat Kaustubh Dubale	20BCE0660
Raksha Gupta	20BCE0548
Somay Vaidh	20BCE2805
Anmol Srivastava	20BIT0097

1. INTRODUCTION

1.1 Overview

We routinely encounter too much information in the form of social media posts, blogs, news articles, research papers, and other formats. This represents an infeasible quantity of information to process, even for selecting a more manageable subset. The process of computationally pruning any document into a shorter version in a manner that preserves as much information as possible and still conveys the overall idea of the original document is Text Summarization. *Text summarization* is an active subfield of *natural language processing* (NLP). Two popular approaches used today to generate automatic text summarization are *extractive text summarization* and *abstractive text summarization*. In extractive text summarization, based on their statistical and linguistic features, sentences or paragraphs are extracted from a source document and concatenated. Whereas in abstractive text summarization, advanced NLP techniques are used to understand the source text and generate a new shorter text that conveys the most pertinent information of the source text.

1.2 Purpose

The purpose of this project is to implement an *extractive text summarization* system. This system when given a large input text data will generate an extractive summary.

2. LITERATURE SURVEY

2.1 Existing problem

Extractive text summarization is a well-studied problem in natural language processing and information retrieval. The key challenge is to identify the most important sentences in a text while maintaining the coherence and relevance of the summary. Existing approaches have focused on various methods, including frequency-based, graph-based, statistical, and machine learning techniques, to address this problem.

2.2 Proposed solution

The proposed solution in this project combines three popular extractive text summarization models: BERT, GPT2, and XLNET. These models have been widely used and proven effective in generating extractive summaries.

- bert_summary: This function utilizes the BERT (Bidirectional Encoder Representations from Transformers) model for text summarization. BERT is a transformer-based model that learns contextual representations of words by considering their surrounding words. The bert_summary function takes a document as input and generates a summary by extracting the most important information from the text.

- gpt2_summary: The gpt2_summary function utilizes the GPT-2 (Generative Pre-trained Transformer 2) model for text summarization. GPT-2 is a large-scale transformer-based language model that is trained to generate coherent and contextually relevant text. In this function, GPT-2 is fine-tuned for summarization tasks, taking a document as input and generating a summary by generating new sentences that capture the essence of the original text.

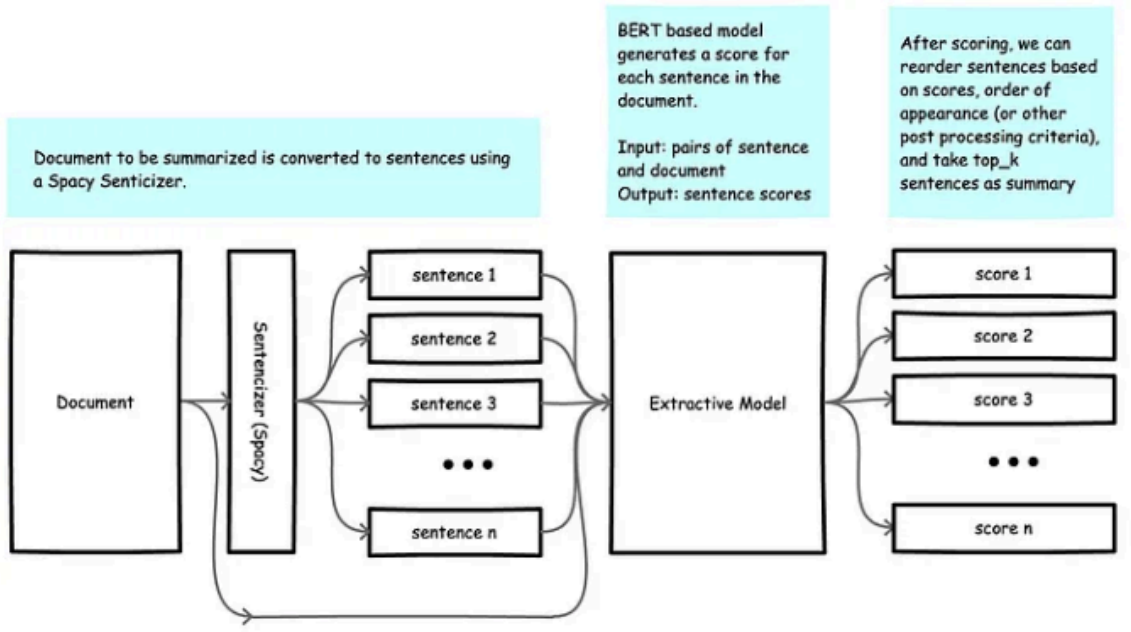
- xlnet_summary: The xlnet_summary function uses the XLNet (eXtreme Learning Machine Network) model for text summarization. XLNet is another transformer-based language model that overcomes some limitations of previous models by considering all possible permutations of words during training. The xlnet_summary function applies XLNet to the input document and generates a summary by

extracting the most important information, similar to the other functions.

These functions leverage powerful pre-trained models to automatically summarize text, providing concise representations of the original content..

3. THEORETICAL ANALYSIS

3.1 Block diagram



Text summarization inference with an extractive summarization model.

3.2 Hardware / Software designing

Hardware Requirements:

- Computer or Server: A computer or server capable of running Python and Flask web framework.
- Processor: A processor with a reasonable speed to handle the computational requirements.
- Memory: Sufficient RAM to accommodate the running processes and store necessary data.
- Storage: Adequate storage space to store the Python code, libraries, and any additional resources.

Software Requirements:

- Python: The project requires Python programming language to run the code. Ensure that Python is installed on the system.
- Flask: Install the Flask web framework to create and run the web application.
- Libraries: Install the necessary Python libraries, including spacy, nltk, sumy, BeautifulSoup, and any other dependencies used in the code. These libraries can be installed using the pip package manager.
- Web Browser: Any modern web browser (such as Google Chrome, Mozilla Firefox, or Microsoft Edge) to access and interact with the web application.

4. EXPERIMENTAL INVESTIGATIONS

Here are some of the main aspects that were explored:

- Summarization Algorithms: The project involved a careful analysis of different extractive summarization algorithms, including BERT, GPT2, and XLNET. Each algorithm was studied in terms of its underlying principles, strengths, and limitations. The analysis focused on selecting algorithms that can effectively capture the most important sentences from the text while maintaining coherence and relevance in the generated summaries.

- Natural Language Processing (NLP): The solution required the use of NLP techniques to preprocess and analyze the text data. The investigation involved exploring libraries and models, such as spaCy, to perform tasks like tokenization, part-of-speech tagging, and named entity recognition. The selection of the appropriate NLP tools was crucial to ensure accurate and meaningful text analysis for the summarization process.

- Evaluation Metrics: To assess the quality of the generated summaries, evaluation metrics were analyzed. Metrics like ROUGE were studied to measure the similarity between the generated summaries and reference summaries. The investigation focused on understanding the

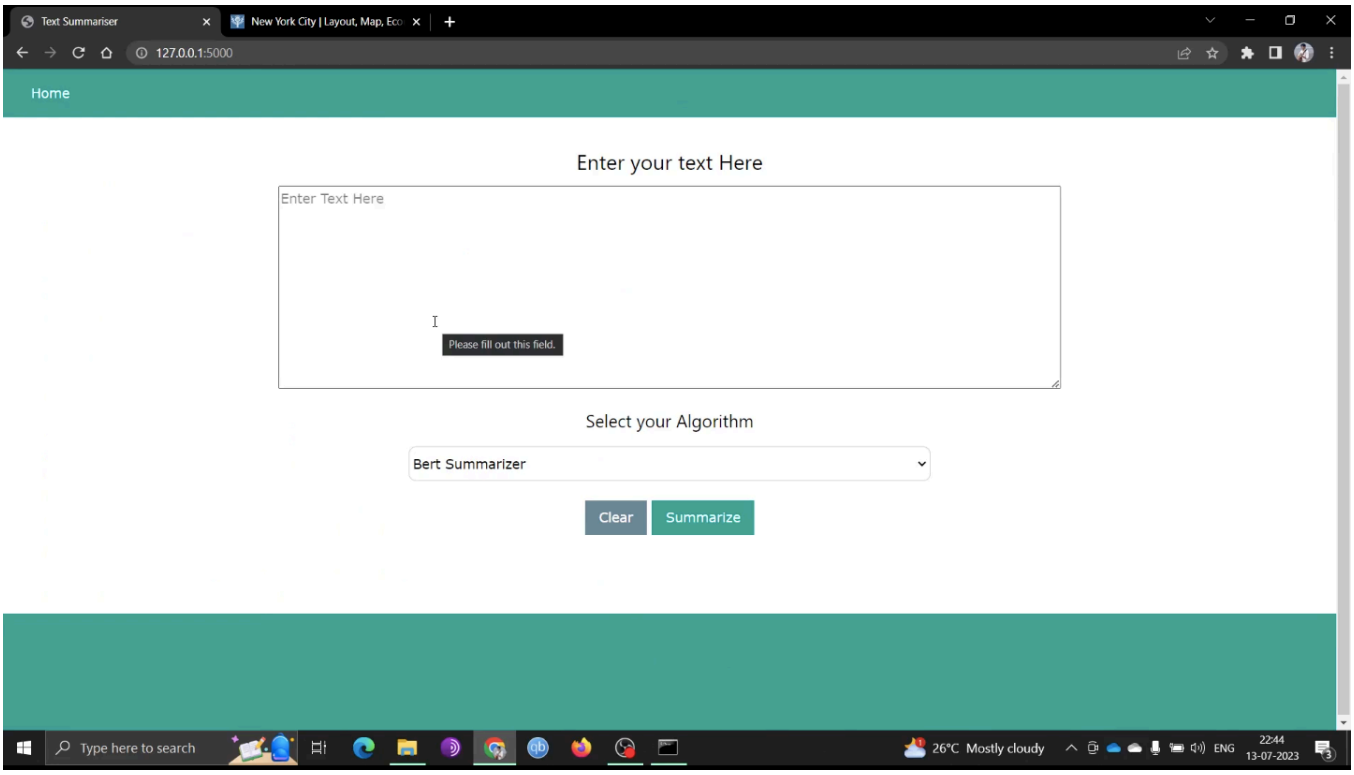
strengths and limitations of these metrics and their applicability to extractive summarization.

- User Experience (UX) Considerations: The investigation also included considerations for user experience. The analysis involved designing a user-friendly interface, optimizing the input process, and providing estimated reading times. This investigation aimed to enhance the usability and overall satisfaction of users interacting with the web application.

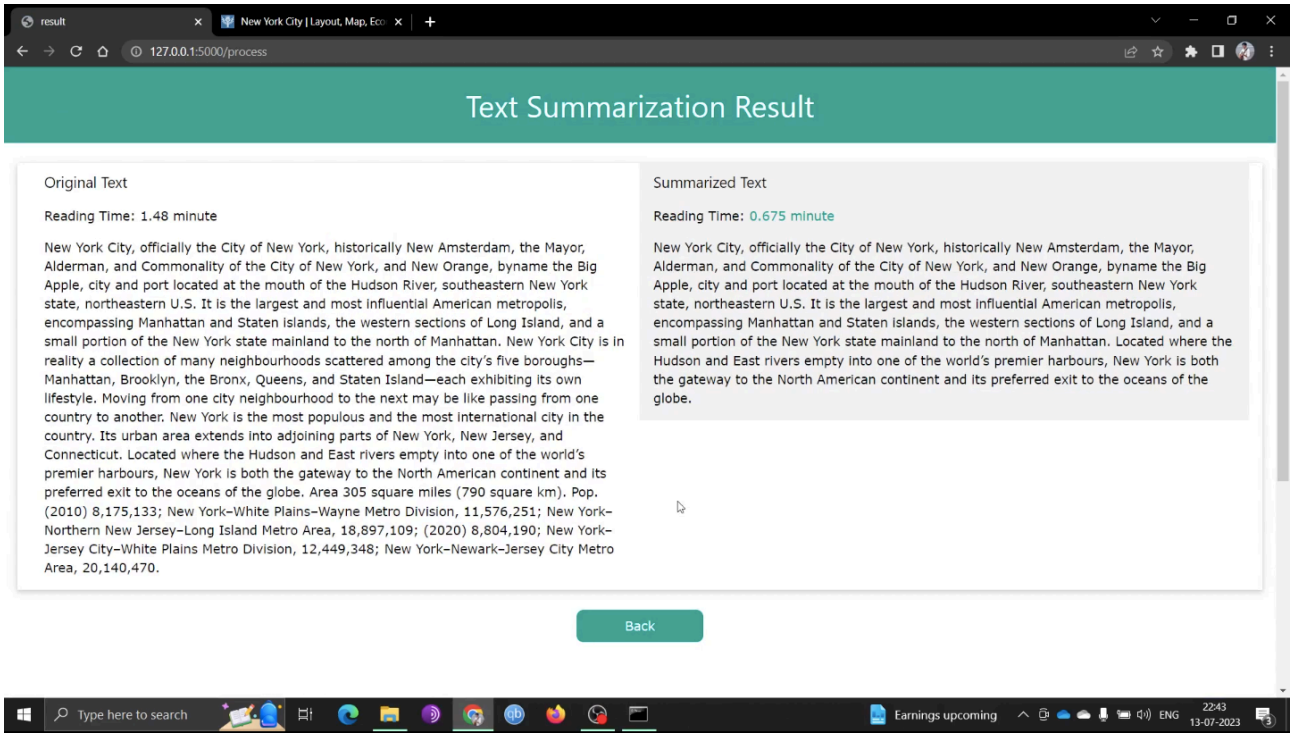
These analyses and investigations played a crucial role in selecting appropriate algorithms, libraries, and techniques for implementing the solution. They helped ensure the solution's accuracy, relevance, user-friendliness, and performance, resulting in a reliable and effective tool for summarizing text content.

5. RESULT

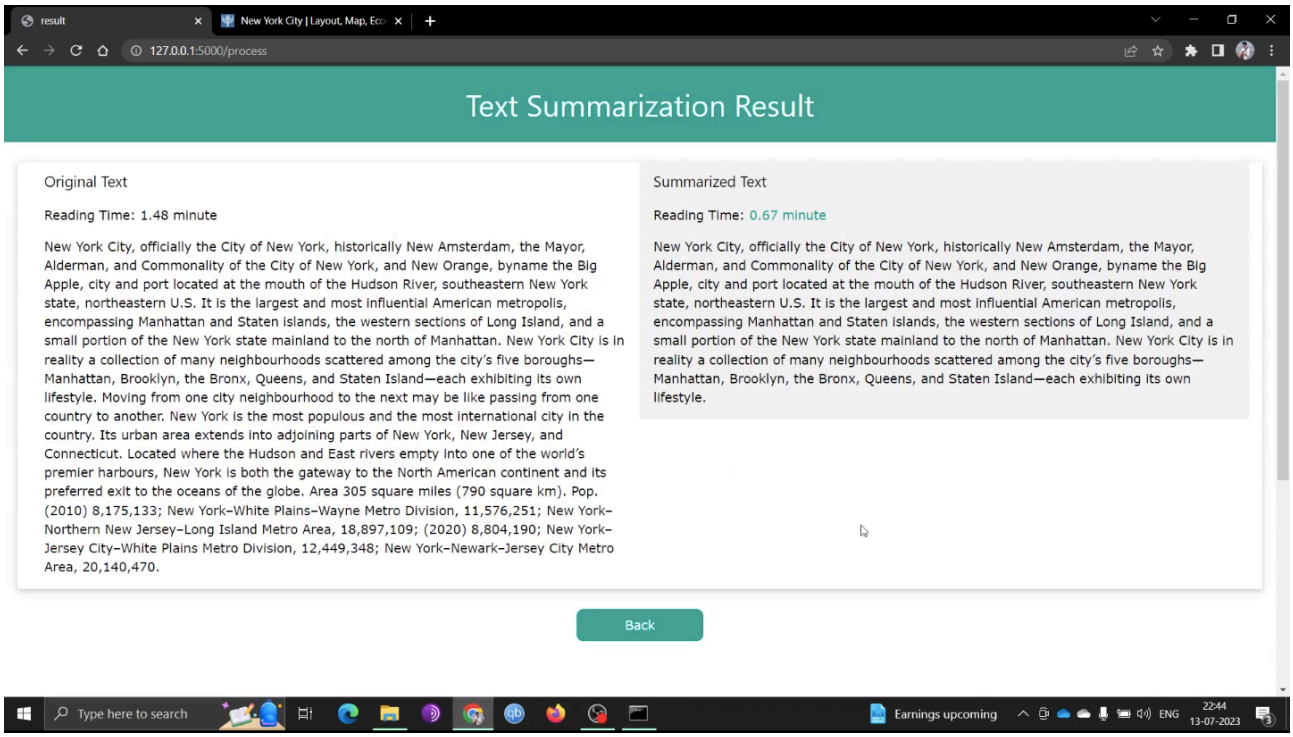
Final findings (Output) of the project along with screenshots.



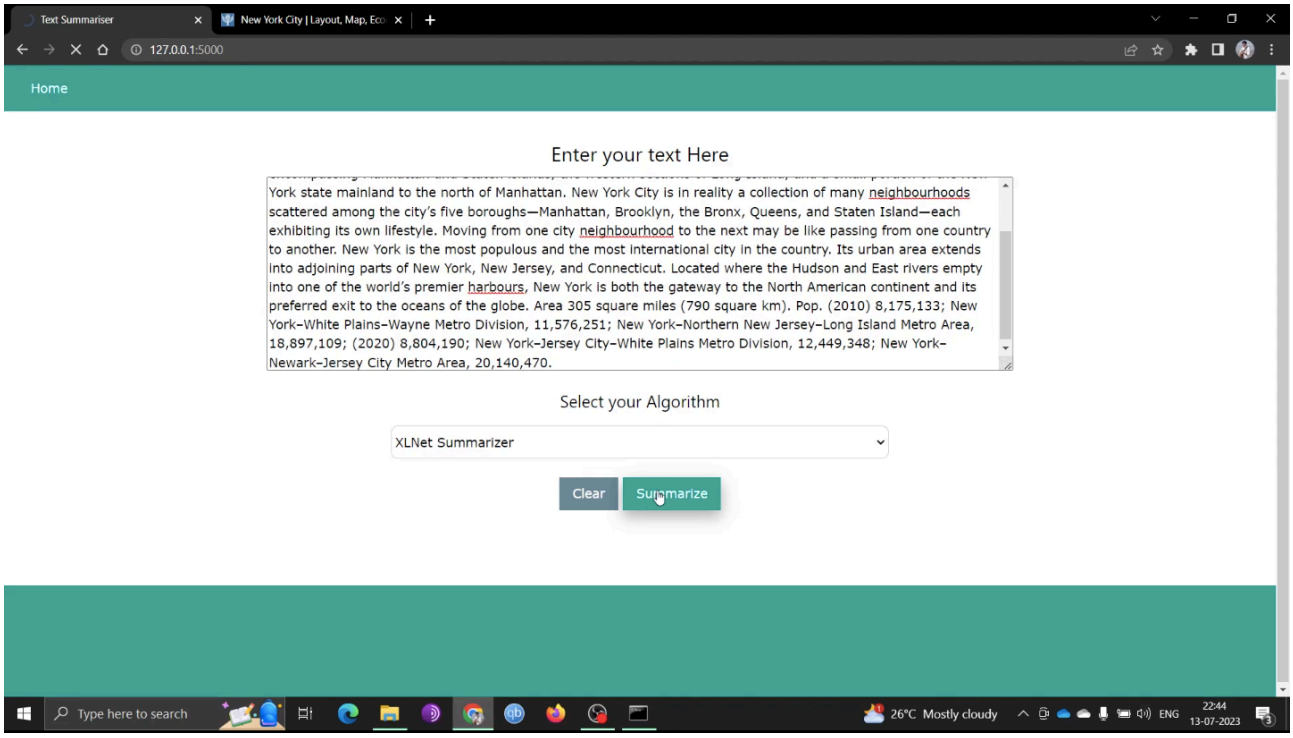
Using Bert Summarizer

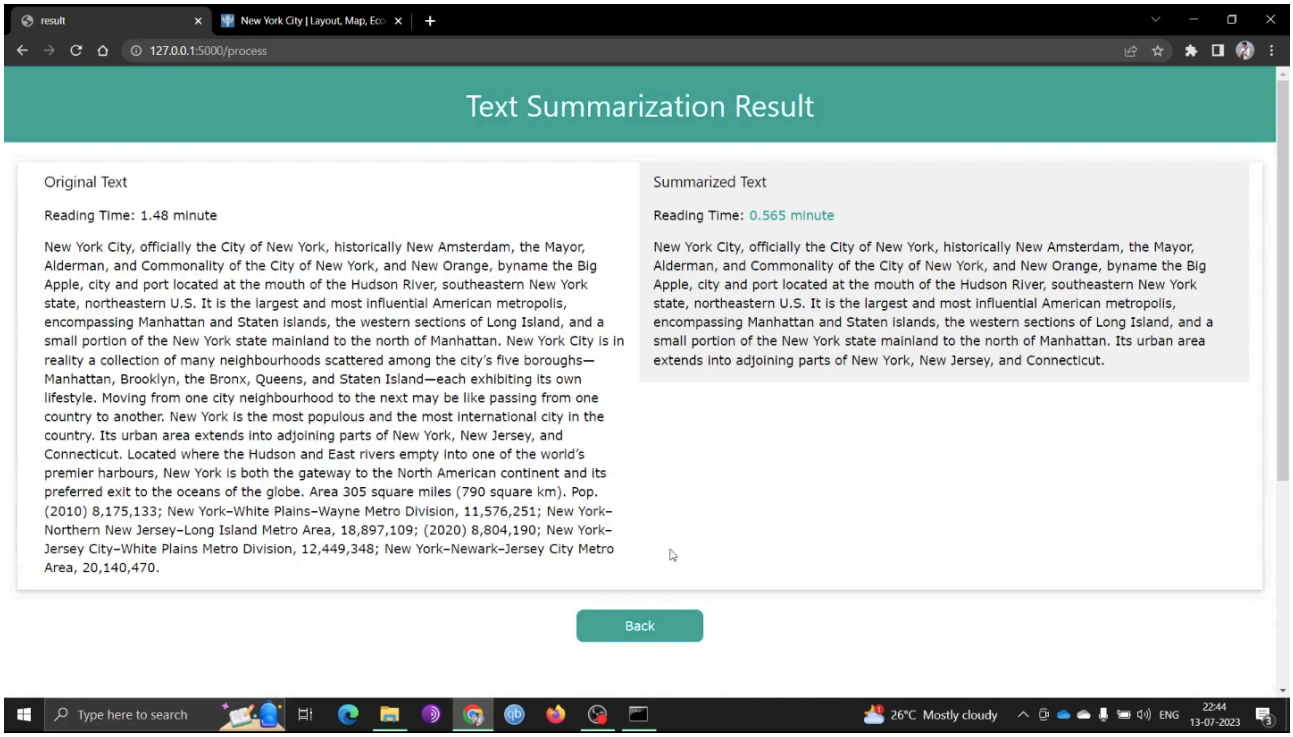


Using GPT2 Summarizer



Using XLNet Summarizer





6. ADVANTAGES & DISADVANTAGES

Advantages of the Proposed Solution:

- Time Efficiency: The extractive text summarization solution allows users to quickly generate summaries from large volumes of text, saving time and effort compared to manually reading and summarizing the entire document.
- Information Retrieval: The generated summaries provide users with a concise overview of the main points and key information within the text. This enables efficient information retrieval and helps users identify relevant content without the need to go through the entire document.
- User-Friendly Interface: The web application offers a user-friendly interface, making it easy for users to input text, select the summarization algorithm, and obtain the generated summary. The estimated reading time feature further assists users in managing their time effectively.
- Algorithmic Flexibility: The solution incorporates multiple summarization algorithms, such as Bert, GPT2, and XLNet. This allows users to choose the algorithm that best suits their requirements and obtain summaries that align with their preferences.
- Scalability: The solution can be applied to various types of text, including user-provided input or text fetched from web pages. This scalability makes it adaptable to different domains and applications where summarization is needed.

Disadvantages of the Proposed Solution:

- Extractive Limitations: Extractive text summarization relies on selecting and rearranging existing sentences from the original text. This approach may result in limitations, as the generated summaries may not capture the complete context or provide original phrasing.
- Redundancy and Repetition: In some cases, the summarization algorithms may select redundant or repetitive sentences, leading to summaries that lack coherence or fail to provide new insights beyond the original text.

- Sensitivity to Input Quality: The quality of the generated summaries is highly dependent on the input text. If the input text is poorly structured, contains grammatical errors, or lacks coherence, the generated summaries may also be of lower quality.

7. APPLICATIONS

This project can be applied in various areas where there is a need to process and summarize large amounts of textual information. Some potential applications include:

- News Aggregation: News websites or applications can use this solution to automatically generate summaries for news articles, allowing users to quickly scan through multiple articles and get a sense of the key information.
- Document Summarization: Organizations dealing with large volumes of documents, such as legal firms or research institutions, can use this solution to summarize lengthy documents, enabling users to quickly identify relevant information without having to read the entire document.
- Content Curation: Content platforms or social media platforms can utilize this solution to summarize user-generated content, blog posts, or articles shared on their platforms. This can help in organizing and presenting content more effectively, enhancing the user experience.
- Market Research: Market research companies can employ this solution to analyze and summarize customer feedback, reviews, and survey responses. Extracting key insights from a large corpus of text can provide valuable information for decision-making and trend analysis.
- E-Learning: Online learning platforms can leverage this solution to summarize lengthy educational materials or textbooks, providing students with concise summaries to aid their understanding and revision.
- Legal Case Analysis: Law firms can utilize this solution to summarize legal cases, judgments, or contracts. This can assist lawyers in

quickly extracting the relevant information and identifying key arguments or clauses.

8. CONCLUSION

In this project, we implemented extractive text summarization using algorithms like bert, gpt2, and xlnet. We developed a Flask web application that allows users to input text or provide a URL for summarization. The summarization algorithms effectively generated concise summaries from the provided text, improving information processing efficiency. The solution can be applied in various domains. Overall, this project demonstrates the usefulness of extractive text summarization in quickly extracting key information from large amounts of text, offering potential benefits in numerous applications.

9. FUTURE SCOPE

While the current implementation is functional, the enhancements that can be made in the future to further improve the solution are adding Abstractive Summarization which involve generating summaries by paraphrasing and rephrasing the original text. Abstractive summarization can provide more fluent and human-like summaries but requires more advanced natural language processing techniques.

We can also add User Feedback and Iterative Improvement which collects user feedback on the generated summaries and iteratively improve the solution based on user suggestions and preferences. This can help in addressing specific user needs and enhancing the overall user experience.

10. APPENDIX

A. Source Code

app.py

```
from flask import Flask,render_template,url_for,request
import time
import spacy
# import nltk
# from sumy.parsers.plaintext import PlaintextParser
# from sumy.nlp.tokenizers import Tokenizer
# from sumy.summarizers.lex_rank import LexRankSummarizer
# from sumy.summarizers.luhn import LuhnSummarizer
# from sumy.summarizers.lsa import LsaSummarizer
from bs4 import BeautifulSoup
from urllib.request import urlopen,Request
from summarizer import Summarizer,TransformerSummarizer

nlp = spacy.load("en_core_web_sm")

app = Flask(__name__)

def bert_summary(docx):
        bert_model = Summarizer()
        result = ''.join(bert_model(docx, min_length=60))
        return result

def gpt2_summary(docx):
        GPT2_model =
TransformerSummarizer(transformer_type="GPT2",transformer_model_key="
gpt2-medium")
        result = ''.join(GPT2_model(docx, min_length=60))
        return result

def xlnet_summary(docx):
```

```
        model =
TransformerSummarizer(transformer_type="XLNet",transformer_model_key="
xlnet-base-cased")
        result = ''.join(model(docx, min_length=60))
        return result
# Reading Time
def readingTime(mytext):
        total_words = len([ token.text for token in nlp(mytext)])
        estimatedTime = total_words/200.0
        return estimatedTime



@app.route('/')
def index():
        return render_template('index.html')

@app.route('/process',methods=['GET','POST'])
def process():
    start = time.time()
    if request.method == 'POST':
        input_text = request.form['input_text']
        model_choice = request.form['model_choice']
        final_reading_time = readingTime(input_text)
        if model_choice == 'bert_summarizer':
            final_summary = bert_summary(input_text)
        elif model_choice == 'gpt2_summarizer':
            final_summary = gpt2_summary(input_text)
        elif model_choice == 'xlnet_summarizer':
            final_summary= xlnet_summary(input_text)
        # elif model_choice == 'isa_summarizer':
        #     final_summary= isa_summary(input_text)
    summary_reading_time = readingTime(final_summary)
    end = time.time()
    final_time = end-start
    return
render_template('result.html',ctext=input_text,final_reading_time=final_reading
```

```
_time,summary_reading_time=summary_reading_time,final_summary=final_s
ummary,model_selected=model_choice)



from bs4 import BeautifulSoup
from urllib.request import urlopen

def get_text(url):
    reqt = Request(url,headers={'User-Agent' : "Magic Browser"})
    page = urlopen(reqt)
    soup = BeautifulSoup(page)
    fetched_text = ' '.join(map(lambda p:p.text,soup.find_all('p')))
    return fetched_text

@app.route('/process_url',methods=['GET','POST'])
def process_url():
        start = time.time()
        if request.method == 'POST':
                input_url = request.form['input_url']
                raw_text = get_text(input_url)
                final_reading_time = readingTime(raw_text)
                final_summary = lex_summary(raw_text)
                summary_reading_time = readingTime(final_summary)
                end = time.time()
                final_time = end-start
        return render_template('result.html',ctext=raw_text,
                        final_summary=final_summary,
                        final_time=final_time,
                        final_reading_time=final_reading_time,
                        summary_reading_time=summary_reading_time)



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

index.html

```
<!DOCTYPE html>
<html>
<head>
        <title>Text Summariser</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/
4.7.0/css/font-awesome.min.css">
<link rel="stylesheet"  href="{{ url_for('static', filename='w3.css') }}">

</style>
</head>
<body>
<div class="w3-bar w3-teal w3-padding">
  <a href="{{url_for('index')}}" class="w3-bar-item w3-button w3-padding
">Home</a>

</div>
<br>

<!-- Start of Main Section -->
<div class="w3-container w3-center"><h3>Enter your text Here</h3>
        <form method="POST" action="/process">
        <textarea  name="input_text" cols="5" rows="10"  required="true"
placeholder="Enter Text Here" style="width:60%"></textarea>
       <br/>

       <p><h4>Select your Algorithm</h4></p>
       <select class="w3-select w3-border w3-round-large"  name="model_choice"
style="width:40%">

            <option value="bert_summarizer"  selected>Bert Summarizer</option>
            <option value="gpt2_summarizer">GPT2 Summarizer</option>
            <option value="xlnet_summarizer">XLNet Summarizer</option>

            </select>
            <br>
            <br>
       <button class="w3-btn w3-blue-grey" type="reset" value="reset">Clear</
button>
        <button class="w3-btn w3-teal" type="submit" value="reset">Summarize</
button>
```

```
        <div class="input-field">
       </div>
</form></div>
<br><br><br><br></div>
</div>
<div class="w3-container w3-teal">
<br><br><br><br><br><br></div>
</body>
</html>
```

result.html

```
<!DOCTYPE html>
<html >
<head>
  <meta charset="UTF-8">
  <title>result</title>
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
  <link rel="stylesheet"  href="{{ url_for('static', filename='w3.css') }}">
  <style type="text/css">
  </style>
</head>
<body>
<div class='w3-padding w3-teal w3-center'><h1>Text Summarization Result</h1></div>
</form>
</div>
</div>

<div class="w3-container w3-padding-24">
<div class="w3-container w3-card">
 <div class="w3-row">
  <div class="w3-half w3-container ">
    <h5>Original Text</h5>
          <p>Reading Time: {{ final_reading_time }} minute </p>
          <p >{{ctext}}</p>
```

```
</div>

  <div class="w3-half w3-container w3-light-grey">
    <h5>Summarized Text</h5>

              <p>Reading Time: <span class="w3-text-
teal">{{ summary_reading_time }} minute </span></p>
            <p>{{ final_summary }}</p>
  </div>
</div>
</div>
</div></div>
<div class="w3-center"><a href="{{url_for('index')}}" style="text-
decoration:none"><button class="w3-btn w3-teal w3-round-large"
style="width:10%;text-decoration:none;" type="submit" value="reset">Back</
button></a></div>
<br><br><br><br><br><br><br><br><br><br><br><br>
<div class="w3-container w3-teal">
<br><br><br><br><br><br><br><br>
</div>
</body>
</html>
```