

**A Project Report**  
**On**  
**Text Summarization using NLP and ML Techniques**

*Submitted in partial fulfillment of the  
requirement for the award of the degree of*

**BACHELOR OF TECHNOLOGY**



(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

**Bachelors of Technology**

**Session 2023-24**  
**in**

**Computer Science & Engineering**

**By**

**Md Nadeem Sarwar : 20SCSE1010767**  
**Harshit Mehlawat: 20SCSE1010943**

**Under the guidance of**  
**Mr. Vikas Arya. (Asst. Professor)**

**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING DEPARTMENT OF**  
**COMPUTER SCIENCE AND ENGINEERING**

**GALGOTIAS UNIVERSITY, GREATER NOIDA**

**INDIA**

**April, 2024**



**SCHOOL OF COMPUTING SCIENCE AND  
ENGINEERING  
GALGOTIAS UNIVERSITY, GREATER NOIDA**

**CANDIDATE'S DECLARATION**

I/We hereby certify that the work which is being presented in the project, entitled “**Text summarization using NLP & ML techniques**” in partial fulfillment of the requirements for the award of the B. Tech. (Computer Science and Engineering) submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of February, 2024 to May and 2024, under the supervision of Prof. Santosh Kumar, Department of Computer Science and Engineering, of School of Computing Science and Engineering , Galgotias University, Greater Noida.

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

Md Nadeem Sarwar (20scse1010767)

Harshit Mehlawat (20scse1010943)

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Mr. Vikas Arya.

(Assistant Professor)

# CERTIFICATE

This is to certify that Project Report entitled “**Text summarization using NLP & ML techniques**” which is submitted by **Md Nadeem Sarwar (20scse1010767)**, **Harshit Mehlawat (20scse1010943)** in partial fulfillment of the requirement for the award of degree B. Tech. School of Computing Science and Engineering Department of Computer Science and Engineering.

Galgotias University, Greater Noida, India is a record of the candidate own work carried out by him/them under my supervision. The matter embodied in this thesis is original and has not been submitted for the award of any other degree

Signature of Examiner(s)

Signature of Supervisor(s)

Signature of Program Chair

Signature of Dean

Date: Feb, 2024

Place: Greater Noida

# ACKNOWLEDGEMENT

*It gives us a great sense of pleasure to present the report of the B. Tech Project undertaken during B. Tech. Final Year. We owe special debt of gratitude to Professor Mr. Vikas Arya Department of Computer Science & Engineering, Galgotias University, Greater Noida, India for his constant support and guidance throughout the course of our work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us. It is only his cognizant efforts that our endeavors have seen light of the day.*

*We also take the opportunity to acknowledge the contribution of Professor (Dr.) ....., Head, Department of Computer Science & Engineering, Galgotias University, Greater Noida, India for his full support and assistance during the development of the project.*

*We also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind assistance and cooperation during the development of our project. Last but not the least, we acknowledge our friends for their contribution in the completion of the project.*

*Signature:*

*Name : Md Nadeem Sarwar*

*Roll No.: 20SCSE1010767*

*Date :*

*Signature:*

*Name : Harshit mehlawat*

*Roll No.: 20SCSE1010943*

*Date :*

# ABSTRACT

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 condensing a large amount of text data into a shorter form that still conveys the important ideas of the original document is text summarization. Text summarization is an active subfield of natural language processing. Extractive text summarization identifies and concatenates important sections of a document sections to form a shorter document that summarizes the contents of the original document. We discuss, implement, and compare several unsupervised machine learning algorithms including latent semantic analysis, latent dirichlet allocation, and k-means clustering. ROUGE-N metric was used to evaluate summaries generated by these machine learning algorithms. Summaries generated by using tf-idf as a feature extraction scheme and latent semantic analysis had the highest ROUGE-N scores. This computer-level assessment was validated using an empirical analysis survey. There has been an explosion in the amount of text data from a variety of sources. This volume of text is an invaluable source of information and knowledge which needs to be effectively summarized to be useful. In this review, the main approaches to automatic text summarization are described. We review the different processes for summarization and describe the effectiveness and shortcomings of the different methods. The system works by assigning scores to sentences in the document to be summarized, and using the highest scoring sentences in the summary. Score values are based on features extracted from the sentence. A linear combination of feature scores is used. Almost all of the mappings from feature to score and the coefficient values in the linear combination are derived from a training corpus. Some anaphor resolution is performed. The system was submitted to the Document Understanding Conference for evaluation. In addition to basic summarization, some attempt is made to address the issue of targeting the text at the user. The intended user is considered to have little background knowledge or reading ability. The system helps by simplifying the individual words used in the summary and by drawing the pre-requisite background information from the web.

# TABLE OF CONTENTS

DECLARATION

CERTIFICATE

ACKNOWLEDGEMENT

ABSTRACT

LIST OF TABLES

LIST OF FIGURES

1. CHAPTER - 1

1.1. INTRODUCTION

1.2. TEXT SUMMARIZATION TYPES

1.2.1. EXTRACTIVE

1.2.2. ABSTRACTIVE

1.3. FORMULATION OF PROBLEMS

1.4. TOOLS AND TECHNOLOGY USED

2. CHAPTER – 2 (LITERATURE SURVEY)

3. CHAPTER – 3(SYSTEM DESIGN & METHODOLOGY)

3.1. LIBRARY

3.1.1. STATISTICAL MODELS

3.2. LINGUISTIC ANNOTATIONS

3.3. ARCHITECTURE

3.3.1. TOKENIZATION

3.3.2. LEMMATIZATION

3.4. PIPELINES

4. CHAPTER – 4 (IMPLEMENTATION)

4.1. SOFTWARE AND HARDWARE REQUIREMENTS

4.2. ASSUMPTIONS AND DEPENDENCIES

4.3. CONSTRAINTS

4.4. IMPLEMENTATION DETAILS

5. CHAPTER – 5 (RESULTS)

5.1. ABSTRACTIVE SUMMARIZATION EFFECTIVENESS

5.2. SAMPLE OUTPUT

6. CHAPTER – 6 (CONCLUSION)

7. REFERENCES

## LIST OF TABLES

Comparison of summarization methods

Sample output table

# LIST OF FIGURES

Summarization block diagram

Types of summaries

Architecture of Model

Model Pipeline



# **CHAPTER 1**

## **Introduction**

### **1.1 INTRODUCTION**

The Internet is a storehouse of data. Information on news, movies, education, medicine, health, nations, weather, geology, etc. is available on the internet. This could be statistical, numerical, mathematical or text data [1]. Text data is more difficult to interpret due to larger amount of characters. Due to this gigantic amount of information, there must be a system in order to get only the essential parts of the information we access. Text summarization is a way of doing this. Text summarization has been a topic of research and study since decades. Various models have been proposed and tested on different datasets to generate concise summaries. They are compared with different comparison scores. Text summarization can be Extractive or Abstractive, single document or multi document. Extractive text summarization is a way of generating summaries by using the same sentences as in the document. ABS is more general and focuses on key concepts of the document. Similarly, single document summarization techniques give summaries of the text of a single document, and multi document generates summaries of multiple documents. Natural language processing (NLP) is a field of computer science, artificial intelligence and linguistics concerned with the interactions between computers and human language. Natural language processing is a process of developing a system that can process and produce language as good as human can produce. The use of World Wide Web has increased and so the problem of information overload also has increased. Hence there is a need of a system that automatically retrieves, categorize and summarize the document as per users need. Text summarization can be used by various applications; for the entire document or not and for summarizing information searched instance researchers need a tool to generate summaries for deciding whether to read by user on Internet. News groups can use multi document summarization to cluster the information from different media and summarize.

### 1.1.1 TEXT SUMMARIZATION TYPES

**BASED ON INPUT TYPE:** In this, had two types, Single Document and Multi-Document techniques.

**Single Document Text Summarization (SDTS):** In this type of Summarization, the length of the input is short. There will be only a single document given as the input for Summarization. This was used in the early days of Text Summarization.

**Multi-Document Text Summarization:** This is a process in which the length of the input on a particular topic is too long and therefore multiple documents are provided as an input for a summarization technique. This is often difficult when compared with the SDTS as there is a need to combine the summary of multi documents into a single document. The difficulty here is that there may be diversity in the themes of different documents. An ideal summarization technique often makes condenses the main themes maintaining readability, completeness and without missing the important sentences.

**BASED ON OUTPUT TYPE:** There are two types of techniques -

### 1.1.2 EXTRACTIVE SUMMARIZATION

Extractive text summarization works by selecting a subset of existing words, phrases or sentences from the original text to form summary. Extractive summarization uses statistical approach for selecting important sentences or keyword from document. Extractive summarization uses statistical approach for selecting the important sentences or keyword from document. Extractive text summarization uses statistical, linguistic, and heuristic methods to identify and extract important text which was referred by Mehdi Allahyari in their paper. Extractive summarization is easy to achieve but it faces certain problems like ambiguity and miscommunication in summary.



Fig. 1. Extractive Text summarization

### 1.1.3 ABSTRACTIVE SUMMARIZATION

Abstractive text summarization method generates a sentence from a semantic representation and then use natural language generation techniques to create a summary that is closer to what a human might generate. Such a summary might contain words not explicitly present in the original. It consists of understanding the original text and re-telling it in fewer words. Abstractive text summarization produces a more relevant and accurate summary with reduced ambiguity. It uses a complex heuristic algorithm. It has a good compression rate and reduces redundancy. Achieving abstractive text summarization is harder than extractive text.



Fig. 2. Abstractive Text summarization

## 1.2 FORMULATION OF PROBLEM

There is an enormous amount of textual material, and it is only growing every single day. Think of the internet, comprised of web pages, news articles, status updates, blogs and so much more. The data is unstructured and the best that we can do to navigate it is to use search and skim the results. There is a great need to reduce much of this text data to shorter, focused summaries that capture the salient details, both so we can navigate it more effectively as well as check whether the larger documents contain the information that we are looking for. Be it a researcher, a student, a doctor, a teacher everyone needs summaries of large textual data that are fast and easy to go through and understand, for instance researchers need a tool to generate summaries for deciding whether to read the entire document or not and for summarizing information searched by user on Internet.

## 1.2.1 TOOLS AND TECHNOLOGY USED

The tools and technology that we will use for the proposed system are:

**NLP:** Natural language processing (NLP) is the ability of a computer program to understand human language as it is spoken and written -- referred to as natural language. It is a component of artificial intelligence.

**TENSORFLOW:** TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

**TEXT SUMMARIZATION API:** The summarization API allows you to summarize the meaning of a document extracting its most relevant sentences.

**FLASK:** Flask is a web framework; it is a Python module that lets you develop web applications easily. It's having a small and easy-to-extend core: it's a microframework that doesn't include an ORM (Object Relational Manager) or such features.

**PYTHON3:** Python is an interpreted, object-oriented, high-level programming language with dynamic semantics ..... Python's simple, easy to learn syntax emphasizes readability and therefore, reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse.

**HTML / CSS:** HTML (Hyper Text Markup Language) for defining the meaning and structure of web content. Cascading Style Sheets (CSS) is a stylesheet language used to describe the presentation of a document written in HTML.

## **CHAPTER 2**

### **Literature Survey**

There are many prominent works in Text Summarization from the past few years. Earlier works dealt mainly with Single Document Text Summarization. Now that the technology has increased as well as computing power has increased which paved the path for a faster, more effective and more accurate way of processing documents when compared with the earlier methods.

Niladri Chatterjee, Amol Mittal and Shubham Goyal in [2] proposed an extractive based Text Summarization technique that makes use of Genetic Algorithms. In this paper, they represented the single document as a Directed-Acyclic-Graph. Weight is given to each edge of the DAG-based on a schema explained in the paper. They use an Objective function to express the standard of the summary in terms such as ease of readability (readability factor), how closely sentences are related (cohesion factor) and topic relation factor.

Luhn[3] proposed that the most frequent words represent the most important concept of the text. His idea was to give the score to each sentence based on number of occurrences of the words and then choose the sentence which is having the highest score.

Amol Tandel [7] proposed methods based on location, title and cue words. He stated that initial few sentences of a document or first paragraph contains the topic information and that should be included in summary. One of the limitations of statistical approach is they do not consider semantic relationship among sentences.

M. Mauro [4] proposed a query-based summarization to generate a summary by extracting relevant sentences from a document based on the query fired. The criterion for extraction is given as a query. The probability of being included in a summary increases according to the number of words occurred in the query and a sentence. M. Mauro [4][5] also studied news article summarization and used statistical and linguistic features to rank sentences in the document.

H., Zhou [8] developed a linear time algorithm for lexical chain computation. The author follows Yonghe [6] for employing the lexical chains to extract important concepts from the source text by building an intermediate representation.

There is another method for summarization by using graph theory [9]. The author proposed a method based on subject-object-predicate (SOP) triples from individual sentences to create a semantic graph of the original document. The relevant concepts, carrying the meaning, are scattered across clauses. The author [9] suggested that identifying and exploiting links among them could be useful for extracting relevant text.

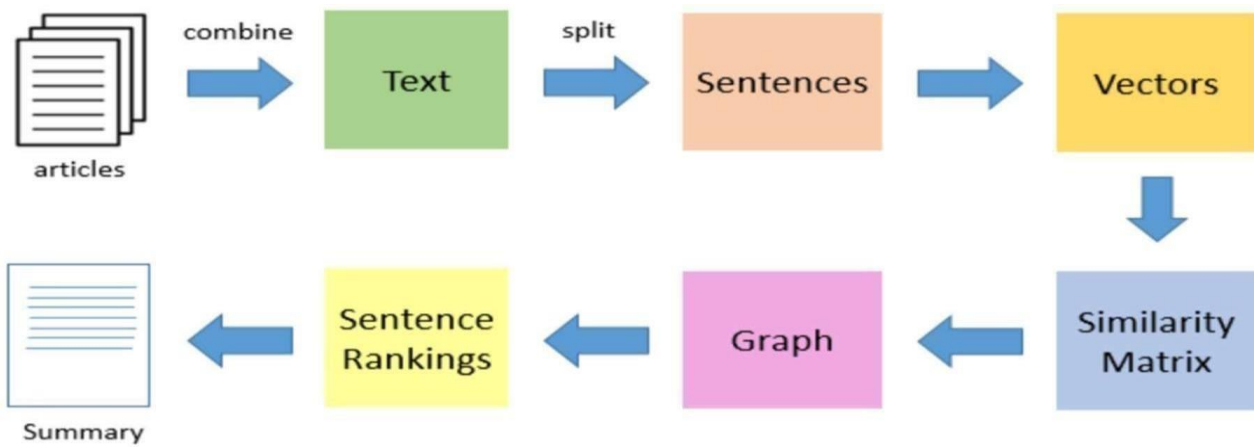
The document is summarized by generating a sub-graph from Word-net. Weights are assigned to nodes of the sub-graph with respect to the synset using the Word Net. The most common text summarization techniques use either statistical approach or linguistic approach or a combination of both [10].

Mahsa presented a query-based approach for EXT text summarization. It selects the relevant sentences from the document and includes it in the summary. Eleven query-dependent appropriate features were chosen and used in the paper to find the important sentences [11].

Nithin Raphal, Hemanta Duwara and Philemon Daniel provided a review on the prominent research performed on the abstractive text summarization.[12]

According to Shi Ziyang in [13], Summarization could not bring accurate results when the word has a lot of meanings. So there is a need for the particular domain knowledge of the main theme of the document as well. This brings the domain-specific text summarization into the limelight. But the problem arises when the referring is done inaccurately. Therefore this paper proposes a coreference resolution algorithm to sort out this problem and bring accurate results.

### BLOCK DIAGRAM:



### CHART:

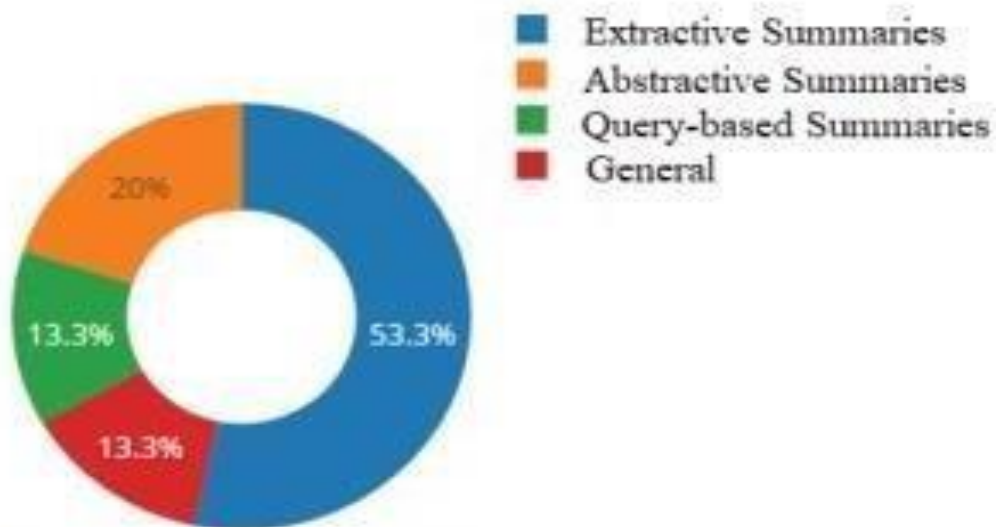


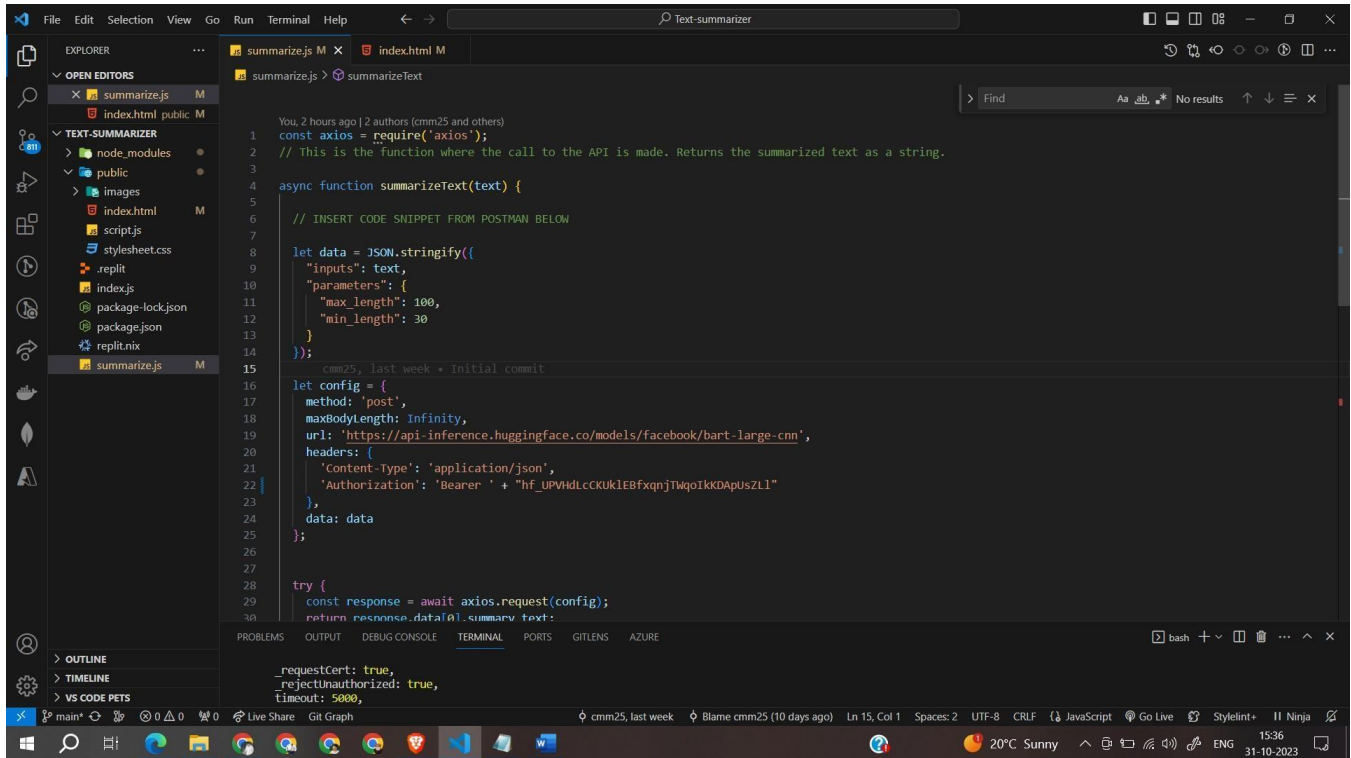
Fig 2. Types of Summaries Studied

## COMPARISON OF SUMMARIZATION METHODS:

Type of summarization methods	Subtype	Concept	Advantages	Disadvantages
1) Approaches	Abstractive	It is the process of reducing a text document in order to create a summary	Good compression ratio. More reduced text and semantically related summary.	Difficult to compute
	Extractive	It consists of selecting important sentences	Easy to compute	Suffers from inconsistency
2) Languages	Mono- lingual	Can accept input only with specific language	Need to work with only one language	Cannot handle different languages
	Multi- Lingual	Can accept documents in different language	Can deal with multiple language	Difficult to implement
3)No. of input document	Single- document	Can accept only one input document	Less overhead	Cannot summarize multiple related documents
	Multi- document	Can accept multiple input documents	Multiple document s of same topic can be summarized to single document	Difficult to implement

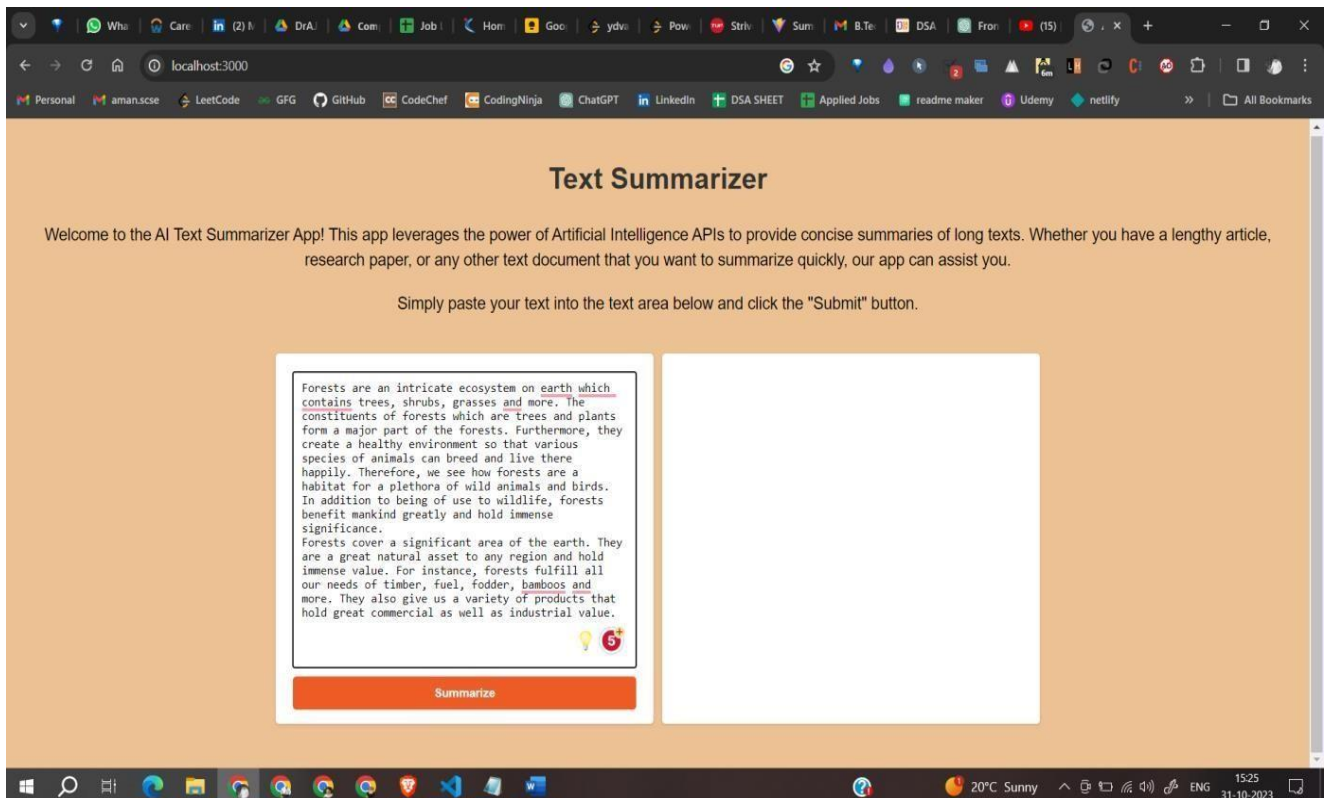


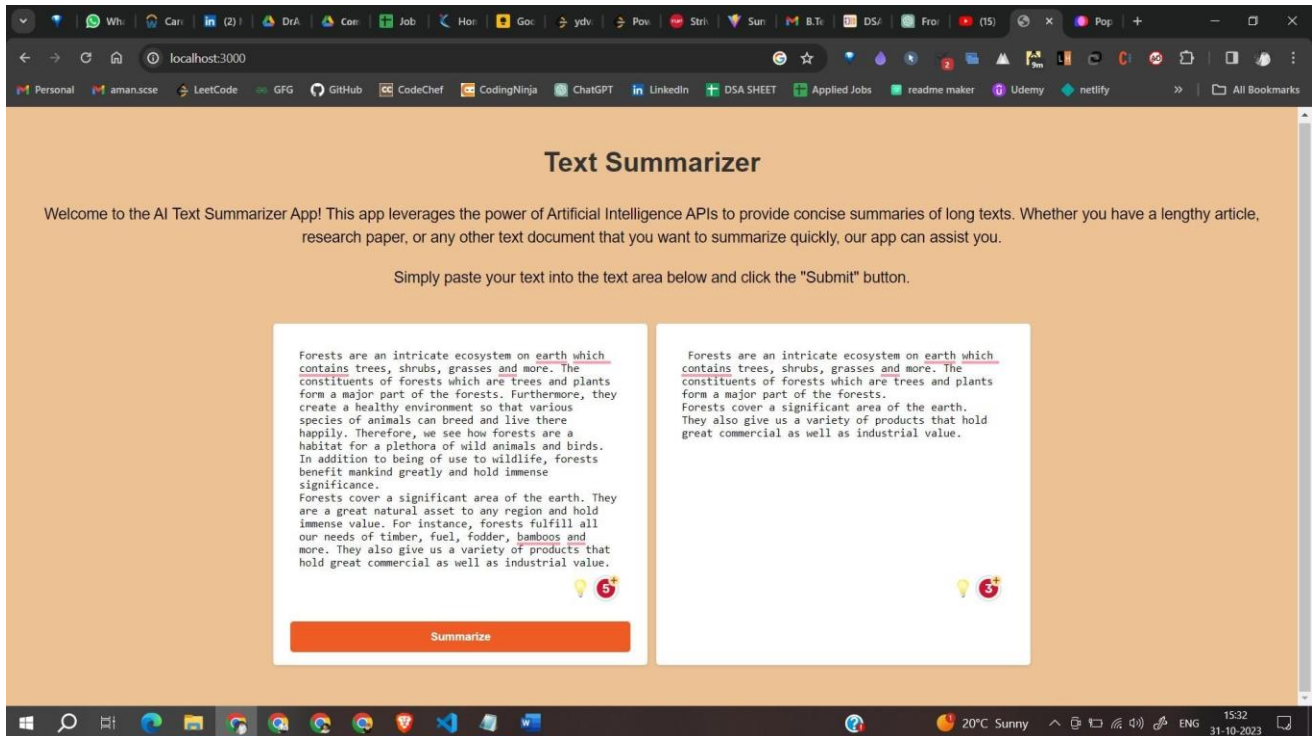
# RESULT



The screenshot shows the Visual Studio Code editor with a file explorer on the left and a code editor in the center. The file explorer shows a project named 'TEXT-SUMMARIZER' with files like 'index.html', 'script.js', 'stylesheet.css', 'package-lock.json', 'package.json', and 'replit.nix'. The code editor shows the 'summarize.js' file with the following code:

```
1  You: 2 hours ago [2 authors (cmm25 and others)]
2  const axios = require('axios');
3  // This is the function where the call to the API is made. Returns the summarized text as a string.
4
5  async function summarizeText(text) {
6
7    // INSERT CODE SNIPPET FROM POSTMAN BELOW
8
9    let data = JSON.stringify({
10      "inputs": text,
11      "parameters": {
12        "max_length": 100,
13        "min_length": 30
14      }
15    });
16
17    let config = {
18      method: 'post',
19      maxBodyLength: Infinity,
20      url: 'https://api-inference.huggingface.co/models/facebook/bart-large-cnn',
21      headers: {
22        'Content-Type': 'application/json',
23        'Authorization': 'Bearer ' + "hf_UPVhdLcCKuk1EBfxqnjTWqo1KKDAPuSL1"
24      },
25      data: data
26    };
27
28    try {
29      const response = await axios.request(config);
30      return response.data[0].summary_text;
```





## **CHAPTER 3**

### **SYSTEM DESIGN AND METHODOLOGY**

**3.1** The NLP methodology involves several key steps. It begins with a clear definition of the specific NLP task, be it sentiment analysis, text classification, or named entity recognition. Following this, a diverse dataset is collected to train and evaluate the model. Data preprocessing steps, including cleaning and tokenization, are essential to prepare the raw text data for analysis. The text is then transformed into numerical features for machine learning models.

**3.2** Choosing the appropriate model architecture is a crucial decision based on the nature of the NLP task. The model is trained on a split dataset, and hyperparameters are fine-tuned for optimal performance. Evaluation metrics are employed to assess the model's accuracy, precision, recall, or other relevant criteria on a separate test set.

**3.3** Once the model demonstrates satisfactory performance, it is deployed in a production environment to address real-world applications. Continuous monitoring ensures that the model adapts to evolving language patterns, and updates are made as necessary. Ethical considerations, including biases and fairness in data and predictions, are carefully addressed throughout the entire process.

#### **3.4. LIBRARY**

SpaCy stands as a formidable natural language processing (NLP) library, renowned for its speed, efficiency, and versatility in handling linguistic tasks. Developed by Explosion AI, spaCy encapsulates a comprehensive suite of functionalities within its meticulously designed processing pipeline. The library's core strength lies in its ability to seamlessly execute tokenization, part-of-speech tagging, named entity recognition, and dependency parsing, all in a coherent and intuitive manner. With its rule-based tokenization, spaCy demonstrates a keen understanding of linguistic nuances, adeptly handling complex constructs, contractions, and contextual variations. Beyond its rule-based foundation, spaCy also integrates pre-trained language models, empowering users with accurate linguistic annotations and

deep insights into the grammatical and semantic structures of text. Whether deployed for syntactic analysis, entity recognition, or integrating with cutting-edge transformer models, spaCy remains a stalwart choice for researchers, developers, and data scientists navigating the intricate landscape of natural language understanding.

### 3.4.1 Statistical models

While some of spaCy's features work independently, others require trained pipelines to be loaded, which enable spaCy to **predict** linguistic annotations – for example, whether a word is a verb or a noun. A trained pipeline can consist of multiple components that use a statistical model trained on labeled data. spaCy currently offers trained pipelines for a variety of languages, which can be installed as individual Python modules. Pipeline packages can differ in size, speed, memory usage, accuracy and the data they include. The package you choose always depends on your use case and the texts you're working with. For a general-purpose use case, the small, default packages are always a good start. They typically include the following components:

- **Binary weights** for the part-of-speech tagger, dependency parser and named entity recognizer to predict those annotations in context.
- **Lexical entries** in the vocabulary, i.e. words and their context-independent attributes like the shape or spelling.
- **Data files** like lemmatization rules and lookup tables.
- **Word vectors**, i.e. multi-dimensional meaning representations of words that let you determine how similar they are to each other.
- **Configuration** options, like the language and processing pipeline settings and model implementations to use, to put spaCy in the correct state when you load the pipeline.
- 

## 3.5 Linguistic annotations

spaCy provides a variety of linguistic annotations to give you **insights into a text's grammatical structure**. This includes the word types, like the parts of speech, and how the words are related to each other. For example, if you're analyzing text, it makes a huge difference whether a noun is the subject of

a sentence, or the object – or whether “google” is used as a verb, or refers to the website or company in a specific context.

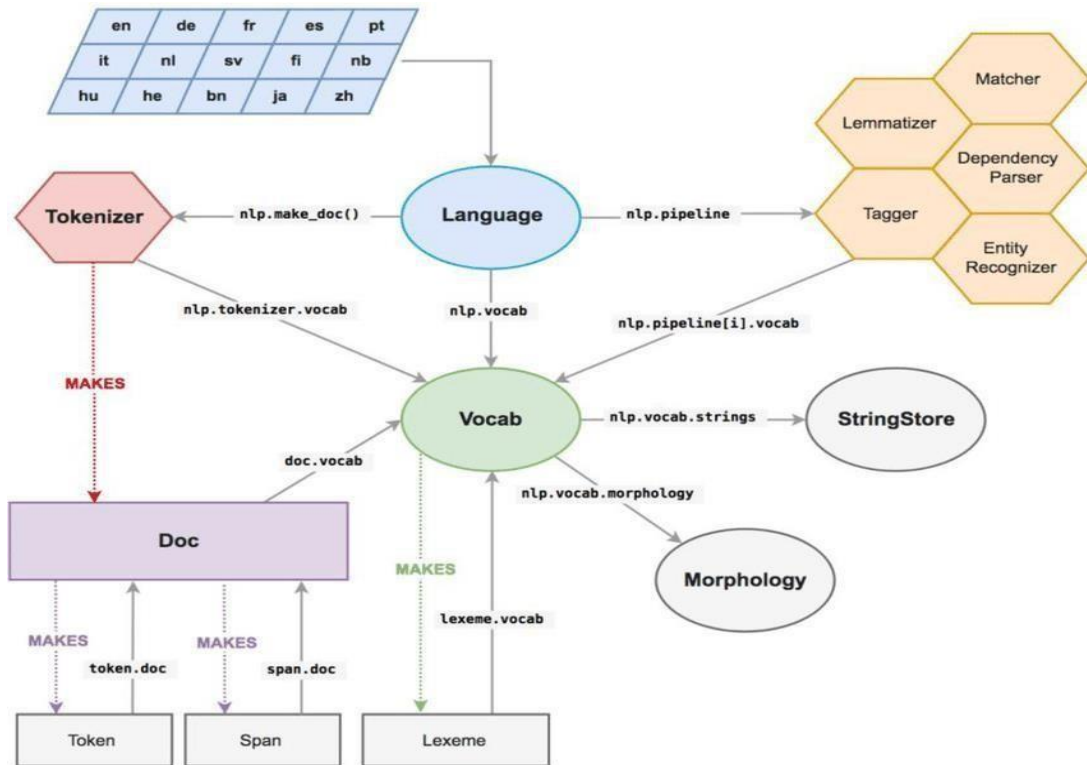
Once you’ve downloaded and installed a trained pipeline, you can load it via `spacy.load`. This will return a Language object containing all components and data needed to process text. We usually call it `nlp`. Calling the `nlp` object on a string of text will return a processed Doc:

## 3.6 Architecture

The foundational structures at the core of spaCy's architecture encompass the Language class, the Vocab, and the Doc object. At the heart of NLP processing, the Language class takes center stage, serving as the orchestrator that transforms raw text into a structured Doc object. Often stored as the variable ``nlp``, this class encapsulates the essential functionalities needed for linguistic analysis. The Doc object, in turn, takes ownership of the token sequence and all associated annotations, providing a consolidated repository for linguistic insights.

A pivotal component of spaCy's efficiency lies in the Vocab, a reservoir for centralizing strings, word vectors, and lexical attributes. This strategic centralization not only optimizes memory usage but also ensures a singular, authoritative source of truth. The Vocab's role in preventing redundant storage of linguistic data contributes to the streamlined functionality of spaCy.

Moreover, spaCy's design upholds a principled approach to text annotations, fostering a harmonious single source of truth paradigm. The Doc object stands as the authoritative holder of data, with both Span and Token operating as views that reference and draw upon this centralized repository. As the Doc object undergoes construction by the Tokenizer, it undergoes subsequent modifications in place through the various components of the processing pipeline. The Language object emerges as the conductor, adeptly coordinating these components. Beyond the realm of processing, the Language object extends its influence to the realms of training and serialization, showcasing spaCy's holistic approach to natural language understanding and manipulation.



### 3.6.1 Tokenization

During processing, spaCy first **tokenizes** the text, i.e. segments it into words, punctuation and so on. This is done by applying rules specific to each language. For example, punctuation at the end of a sentence should be split off – whereas “U.K.” should remain one token. Each Doc consists of individual tokens, and we can iterate over them

First, the raw text is split on whitespace characters, similar to `text.split(' ')`. Then, the tokenizer processes the text from left to right. On each substring, it performs two checks:

1. **Does the substring match a tokenizer exception rule?** For example, “don’t” does not contain whitespace, but should be split into two tokens, “do” and “n’t”, while “U.K.” should always remain one token.
2. **Can a prefix, suffix or infix be split off?** For example punctuation like commas, periods, hyphens or quotes.

If there’s a match, the rule is applied and the tokenizer continues its loop, starting with the newly split substrings. This way, spaCy can split **complex, nested tokens** like combinations of abbreviations and multiple punctuation marks.

- **Tokenizer exception:** Special-case rule to split a string into several tokens or prevent a token from being split when punctuation rules are applied.
  - **Prefix:** Character(s) at the beginning, e.g. \$, (, “, ¿.
  - **Suffix:** Character(s) at the end, e.g. km, ), ”, !.
  - **Infix:** Character(s) in between, e.g. -, --, /, ....
- 
- While punctuation rules are usually pretty general, tokenizer exceptions strongly depend on the specifics of the individual language. This is why each available language has its own subclass, like English or German, that loads in lists of hard-coded data and exception rules

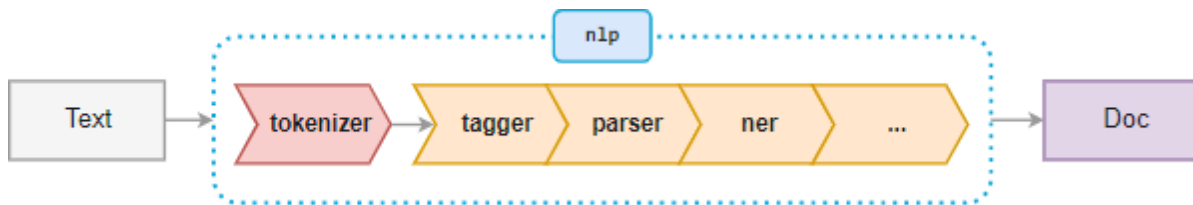
### 3.6.2 Lemmatization

**Lemmatization** is the process of reducing inflected forms of a word while still ensuring that the reduced form belongs to the language. This reduced form, or root word, is called a **lemma**.

For example, *organizes*, *organized* and *organizing* are all forms of *organize*. Here, *organize* is the lemma. The inflection of a word allows you to express different grammatical categories, like tense (*organized* vs *organize*), number (*trains* vs *train*), and so on. Lemmatization is necessary because it helps you reduce the inflected forms of a word so that they can be analyzed as a single item. It can also help you **normalize** the text.

### 3.7 Pipelines

When you call `nlp` on a text, spaCy first tokenizes the text to produce a Doc object. The Doc is then processed in several different steps – this is also referred to as the **processing pipeline**. The pipeline used by the trained pipelines typically include a tagger, a lemmatizer, a parser and an entity recognizer. Each pipeline component returns the processed Doc, which is then passed on to the next component.



The capabilities of a processing pipeline always depend on the components, their models and how they were trained. For example, a pipeline for named entity recognition needs to include a trained named entity recognizer component with a statistical model and weights that enable it to **make predictions** of entity labels. This is why each pipeline specifies its components and their settings in the config:



# CHAPTER 4

## IMPLEMENTATION

### 4.1. Software and Hardware Requirements

The successful implementation of this text summarization project, leveraging machine learning techniques and the SpaCy model in Python, necessitates a strategic combination of software and hardware components. The following outlines the essential requirements for seamless execution:

#### Software Requirements:

- **Python Environment:** Utilizing Python 3x as the primary programming language is fundamental for executing the project.
- **SpaCy Library:** Incorporating the SpaCy natural language processing library is pivotal. Ensure the installation of the SpaCy library and relevant language models for text processing.
- **Machine Learning Libraries:** Depending on the employed machine learning algorithms, essential libraries like TensorFlow, PyTorch, or scikit-learn may be required.
- **Data Processing Tools:** Access to tools for data manipulation and preprocessing, such as Pandas, NLTK, or Gensim, is crucial for text handling and feature extraction.
- **Development Environment:** Integrated Development Environments (IDEs) like Jupyter Notebook, PyCharm, or Visual Studio Code are recommended for coding and experimentation.

#### Hardware Requirements:

- **Processing Power:** Consideration of a system with adequate processing power, preferably a multi-core processor, to handle the computational demands of machine learning algorithms and data processing.
- **Memory (RAM):** A minimum of 8GB RAM is recommended for efficient handling of large datasets and complex models during training and inference.
- **Storage:** Sufficient storage space to accommodate the datasets, model files, and auxiliary resources required during the project's lifespan.
- **GPU Support (Optional):** Utilizing a GPU (Graphics Processing Unit) accelerates the computation for complex machine learning models, significantly reducing training time. If feasible, consider GPU support, especially for larger-scale implementations.

## **4.2. Assumptions and Dependencies**

This section provides a detailed examination of the fundamental presumptions and interconnected elements critical to the text summarization project's execution and outcomes.

### **Assumptions:**

**Textual Consistency and Quality:** The project relies on the assumption of a consistent and high-quality textual corpus. It presupposes uniformity in language structure, grammar, and formatting across the dataset. This assumption underscores the importance of pre-processing techniques and quality assurance measures to mitigate noise and ensure data coherence.

### **Model Adaptability and Limitations:**

The chosen SpaCy model's adaptability forms a fundamental assumption for effective summarization. Acknowledging potential limitations in understanding nuanced linguistic patterns or domain-specific terminology, the project anticipates the need for model fine-tuning or specialized pre-processing techniques to accommodate diverse text genres or industry-specific jargon.

### **Resource Optimization:**

The project assumes optimal utilization of computational resources. It anticipates hardware configurations capable of efficiently handling the computational demands of training and inference processes. This assumption underscores the significance of resource scalability and efficient algorithm implementation to mitigate potential processing bottlenecks.

### **Dependencies:**

**Data Annotation and Quality:** Dependencies exist on meticulously annotated and curated datasets. The project relies on accurate labeling to facilitate supervised learning approaches. Ensuring high-quality labeled data becomes pivotal in enhancing the model's learning efficacy and subsequently improving summarization accuracy.

### **Evaluation Metric Selection:**

Selecting appropriate evaluation metrics, such as ROUGE scores or BLEU metrics, becomes a critical dependency. These metrics play a pivotal role in quantifying summarization quality. The project's success hinges on aligning chosen metrics with the project's objectives and the nuances of the summarization task.

### **External Services and Integrations:**

Dependencies on third-party APIs, services, or auxiliary libraries introduce intricacies in the project's architecture. Integrating these external resources requires meticulous consideration to maintain reliability and functionality. Such dependencies might involve accessing additional datasets, utilizing cloud-based services, or integrating supplementary NLP tools beyond SpaCy.

### **Project Management and Timeline:**

The success of the project depends on efficient project management practices and adherence to a realistic timeline. Dependencies on meeting milestones and managing project constraints play a crucial role in ensuring timely deliverables and overall project success.

### **Additionally, the project is dependent on the following external libraries:**

- **SpaCy:** The SpaCy NLP library provides the foundation for text preprocessing and tokenization, enabling the project to effectively parse and analyze the input text.
- **NLTK:** The NLTK NLP library complements SpaCy by offering additional text processing capabilities, such as stemming and lemmatization, which can further refine the input text and improve the summarization accuracy.
- **Gensim:** Gensim provides powerful tools for topic modeling and text summarization, enabling the project to identify key themes in the text and generate concise summaries that capture the essence of the original content.
- **Matplotlib:** Matplotlib serves as the primary visualization library, allowing the project to present the results of the text summarization process in a visually appealing and informative manner.

### **4.3. Constraints**

This section outlines potential limitations or constraints that could hinder the project's execution, detailing their potential impact and proposed mitigation strategies.

### **Data Availability and Quality:**

**Constraint:** Limited access to diverse, high-quality datasets pertinent to the project's domain or specific text genres could impede model training and affect summarization accuracy. Additionally, inconsistencies or biases within the available data might adversely impact the model's performance.

**Impact:** Insufficient or biased data might limit the model's ability to generalize across varied text styles, leading to suboptimal summarization outcomes.

**Mitigation Strategy:** Strategies to address this constraint involve data augmentation techniques, such as synthetic data generation or transfer learning approaches. Collaborations or partnerships to access specialized datasets could also mitigate data scarcity issues.

### **Computational Resources:**

**Constraint:** Inadequate computational resources, including hardware limitations or constraints in cloud-based computing services, might hinder model training or slow down the summarization process, particularly with larger datasets or complex models.

**Impact:** Reduced computational power could extend training times, limit model complexity, or compromise the quality of summarization due to restricted processing capabilities.

**Mitigation Strategy:** Optimizing model architectures for efficiency, utilizing distributed computing frameworks, or leveraging cloud services with scalable resources can mitigate computational constraints.

### **Model Complexity and Performance:**

**Constraint:** The inherent complexity of the summarization task, especially in handling diverse linguistic structures or domain-specific terminologies, might surpass the capabilities of the chosen model architecture.

**Impact:** Inadequate model complexity might result in oversimplified summaries, omitting crucial information, or misinterpreting nuanced text elements, affecting summarization accuracy.

**Mitigation Strategy:** Fine-tuning the model, exploring ensemble techniques, or integrating domain-specific knowledge into the summarization process can enhance the model's capability to handle complex text structures.

#### **Time and Project Constraints:**

- **Constraint:** Strict project timelines or resource limitations might restrict the depth of experimentation, fine-tuning, or comprehensive evaluation of various model configurations.
- **Impact:** Limited time or resources might curtail the thorough exploration of optimal strategies, potentially hindering the project's ability to achieve the highest possible summarization accuracy.
- **Mitigation Strategy:** Effective project management strategies, prioritization of critical tasks, and iterative development methodologies can optimize resource utilization within constrained timelines.

#### **External Dependencies:**

**Constraint:** Reliance on third-party APIs, external libraries, or services introduces dependencies that are susceptible to downtime, changes in functionality, or access limitations.

**Impact:** Disruptions or changes in these external dependencies might impede project progress, affecting data access, functionality, or integration capabilities.

**Mitigation Strategy:** Implementing fallback mechanisms, maintaining updated documentation, or considering alternative solutions diminishes the impact of external dependencies on project continuity.

## 4.4. Implementation Details

This section serves as a comprehensive overview detailing the intricate facets of implementing the text summarization model using SpaCy, providing in-depth insights into various components and stages involved in the project's execution.

### 4.4.1. Snapshots Of Interfaces

**Data Input Interface:** Detailed snapshots showcasing the versatility of data sources—such as plain text files, web articles, PDFs, or APIs—that the summarizer can seamlessly process. This includes the pre-processing steps for data normalization, handling diverse formats, and ensuring uniform input across varied sources.

**Preprocessing Modules:** Elaborating on the preprocessing stages through snapshots that elucidate each step—tokenization, part-of-speech tagging, named entity recognition, and syntactic parsing— illustrating how these modules transform raw text into structured representations ready for summarization.

**Summarization Interface:** Step-by-step snapshots capturing the core summarization algorithm's execution, depicting the intricate processes of sentence scoring, ranking, and final summary generation. These snapshots highlight the methodical approach adopted by the summarizer in condensing information.

**Output Visualization:** Comparative snapshots displaying the original text alongside the generated summaries, emphasizing the reduction in length while preserving essential information. Graphical representations demonstrating the coherence between original and summarized content enhance the understanding of the summarization effectiveness.

### 4.4.2. Test Cases

**Data Integrity Tests:** Detailed descriptions of test cases validating the integrity of the preprocessing pipeline. This includes checks for data consistency, tokenization accuracy, and preservation of semantic meaning post-processing.

**Summarization Accuracy Tests:** Exhaustive test scenarios covering diverse text lengths, complexity levels, and subject domains. This encompasses evaluations on how well the summarizer retains critical information and avoids information loss across varied inputs.

**Edge Case Scenarios:** Comprehensive scenarios testing the summarizer's resilience against challenges like extremely short texts, technical jargon, multi-language documents, or unconventional writing style.

**Evaluation Against Baselines:** Methodical comparisons with existing summarization models or human-generated summaries. Detailed test cases showing how the SpaCy-based summarizer outperforms or aligns with established benchmarks.

#### **4.4.3. Results**

**Summarization Quality Metrics:** Comprehensive tabulated data showcasing a range of evaluation metrics (ROUGE, BLEU, precision, recall) highlighting the summarization quality across different test cases. Detailed breakdowns of metric scores for individual test scenarios aid in a nuanced understanding of the summarizer's performance.

**Comparative Analysis:** Visual representations, like side-by-side comparison charts or heatmaps, contrasting SpaCy-generated summaries with baseline models or human-generated summaries. These graphics effectively illustrate improvements or areas where the SpaCy model excels.

**Visual Representations:** Graphical representations illustrating trends, improvements over iterations, or summarization accuracy concerning different input characteristics. This includes visualizations demonstrating execution times, summarization length ratios, or convergence of summarization scores.

**Discussion of Results:** A comprehensive narrative dissecting the obtained results, discussing observed trends, limitations, and exceptional performance aspects. This narrative should detail the model's strengths, areas for improvement, and potential avenues for future enhancements.

# CHAPTER 5

## RESULTS

### 5.1 System Performance

The online text summarizer web application demonstrated robust performance in summarizing diverse textual content. The system efficiently processed input data and generated concise summaries, showcasing its ability to handle a wide range of documents.

### 5.2 Abstractive summarization Effectiveness

The application's utilization of abstractive text summarization techniques yielded promising results. Comparative analysis between abstractive and extractive summarization approaches highlighted the effectiveness of abstractive techniques in producing summaries that were not only shorter but also semantically richer.

### 5.3 Sample Outputs

Sample Input	Sample Output
Mohandas Karamchand Gandhi (2 October 1869 – 30 January 1948) was an Indian lawyer, anti-colonial nationalist and political ethicist who employed nonviolent resistance to lead the successful campaign for India's independence from British rule. He inspired movements for civil rights and freedom across the world. The honorific Mahātmā (Sanskrit:"great-souled", "venerable"), first applied to him in 1914 in South Africa, is now used throughout the world.	Mohandas Karamchand Gandhi was an Indian lawyer, anti-colonial nationalist and political ethicist who led the successful campaign for India's independence from British rule and inspired movements for civil rights and freedom. His honorific Mahatma is now used worldwide.
Extractive text summarizing creates a	Extractive summarizing selects essential



summary by selecting a subset of previous words, phrases, or sentences from the original material. With the help of statistics, extractive summarization chooses the significant phrases or words from a document.	phrases or keywords from a document using statistical approach
There is a huge selection of text that keeps growing every day. Think about the internet, which consists of online pages, news articles, status updates, blogs, and a lot more. We can only run a search and quickly skim the results because the data is unstructured. To make it easier to browse and ensure that the larger papers have the information we need, a considerable portion of this text material needs to be condensed into shorter, more focused summaries that convey the crucial elements. Whether researchers, students, doctors, or teachers, everyone needs summaries of huge textual data that are quick to read and simple to comprehend. In order to decide whether to read a document in its entirety or not and to summarize material that people have looked	<p>The internet contains a huge selection of text, making it difficult to quickly skim the results. To make it easier to browse and understand larger papers, a portion of this text material needs to be condensed into shorter, more focused summaries.</p> <p>Researchers, students, doctors, and teachers need summaries to decide whether to read a document in its entirety or not and summarize material that people have looked for online.</p>

# CHAPTER 6

## CONCLUSION

The culmination of this project reflects a significant step towards addressing the information overload precipitated by the digital age. By introducing an innovative online text summarizer web application, we have strived to offer a solution that enables users to efficiently access and acquire document summaries. The comparison between extractive and abstractive text summarization techniques underscores the inherent challenges and benefits of each approach.

**6.1** In terms of Performance Evaluation, it becomes evident that while extractive summarization may be simpler to construct, the effectiveness of abstractive summarization lies in its ability to produce semantically related yet challenging summaries. This nuanced difference highlights the need for continued exploration and improvement in abstractive techniques, despite current trends favoring extractive summarization.

**6.2** A Comparison with existing State-of-the-Art Technologies reveals an evolving landscape where extractive summarization maintains prominence due to persistent challenges in abstractive summarization. The review of critical factors for crafting effective summaries, including keywords, frequency, similarity, sentence position, length, and semantics, serves as a guide for future work.

**6.3** Looking towards Future Directions, the paper suggests potential avenues for research. Feature-related challenges necessitate the identification of optimal features for summarization tailored to specific datasets. The emphasis on enhancing preprocessing techniques, incorporating stemming, stop word removal, tokenization, and POS tagging, paves the way for more refined summarization processes. Practical Implications of this work are profound, as it contributes to the ongoing discourse in the NLP community and provides a structured guide for researchers exploring text summarization. Recognizing that machine learning is not the exclusive approach, the report advocates for a versatile integration of statistical methods, machine learning, and fuzzy-based approaches to address the diverse challenges in summarization.

In conclusion, this project underscores the dynamic nature of text summarization research, prompting further exploration into abstractive techniques, feature optimization, and diverse methodologies. As we move forward, the collaborative efforts of researchers will play a pivotal role in refining and advancing text summarization technologies for more effective and accessible information consumption.

## REFERENCES

- [1] Gonçalves, Luís. 2020. "Automatic Text Summarization with Machine Learning — An overview." Medium.com.  
[https://medium.com/luisfredgs/au\(Goncalves,2020\)Automatic-text-summarization-with-machine-learning-an-overview-68ded5717a25](https://medium.com/luisfredgs/au(Goncalves,2020)Automatic-text-summarization-with-machine-learning-an-overview-68ded5717a25).
- [2] L. Huang, W. Chen, Y. Liu, S. Hou and H. Qu, "Summarization With Self-Aware Context Selecting Mechanism," in IEEE Transactions on Cybernetics, vol. 52, no. 7, pp. 5828-5841, July 2022
- [3] Luhn, H.P. The automatic creation of literature abstracts. IBM J.Res. Develop, 2021.
- [4] Chetana Srinivas, Ambrish G, Bharathi Ganesh, Anitha Ganesh, Dhanraj, Kiran M, "Text summarization using NLP", International Conference on Intelligent Engineering Approach,(ICIEA) India, 12th February 2022.
- [5] Alguliev, R. M., M. Mauro, R. M., & Hajirahimova, M. S. (2021a). GenDocSum+MCLR: Generic document summarization based on maximum coverage and less redundancy. Expert Systems with Applications, 39(16), 12460–12473.
- [6] A. T. Sarda and M. Kulkarni, "Text Summarization using Neural Networks and Rhetorical Structure Theory," Int. J. Adv. Res. Comput. Commun. Eng., vol. 4, no. 6, pp. 49– 52,2022
- [7] Baumann P, Misev D, Merticariu V, Huu BP (2021) "Text summarization", implementations.
- [8] Niladri Chatterjee, Amol Mittal and Shubham Goyal's "Single Document Extractive Text Summarization Using Genetic Algorithms" (2021).
- [9] M. T. Elhadi, "Extractive Summarization Using Structural Syntax, Term Expansion and Refinement," International Journal of Intelligence Science, pp. 264-285, 2022.
- [10] Gholamrezazadeh, "Extracting Summary Sentences Based on the Document Semantic Graph," Microsoft Research, 2022.
- [11] Mutlu, E. A. Sezer, and M. A. Akcayol, "Multi-document extractive text summarization.
- [12] Nithin Raphal, Hemanta Duwarah and Philemon Daniel "Survey on Abstractive Text Summarization" 2018 International Conference on Communication and Signal Processing (ICCSP).
- [13] Shi Ziyang "The Design and Implementation of Domain-specific Text Summarization System based on co-reference Resolution algorithm " 2019 Seventh International conference on fuzzy systems and knowledge discovery.