Text Summarization Using NLP With Deployment on AWS using CI/CD

Major Project Report

Submitted in partial fulfillment of the requirement of University of Mumbai

For the Degree of

Bachelor of Engineering (Computer Engineering)

By

Bhuvan Shingade	**ID No: TU3S2122006**
Suyash Kasar	**ID No: TU3F2021125**
Ved Kolambkar	**ID No: TU3S21220012**
Yash Matha	**ID No. TU3S21220011**

Under the Guidance of

Prof. Rohini Palve

Department of Computer Engineering

TERNA ENGINEERING COLLEGE

Nerul (W), Navi Mumbai 400706

(University of Mumbai)

(2023-2024)

TERNA ENGINEERING COLLEGE, NERUL,

NAVI MUMBAI

Department of Computer Engineering

Academic Year 2023-24

CERTIFICATE

This is to certify that the major project entitles “Text Summarization Using NLP With Deployment on AWS using CI/CD” is a bonafide work of

Bhuvan Shingade	**ID No: TU3S2122006**
Suyash Kasar	**ID No: TU3F2021125**
Ved Kolambkar	**ID No: TU3S21220012**
Yash Matha	**ID No. TU3S21220011**

submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the Bachelor of Engineering (Computer Engineering).

Guide **Head of Department** **Principal**

Project Report Approval

This Major Project I Report – entitled "**Text Summarization Using NLP With Deployment on AWS using CI/CD**" by following students is approved for the degree of ***B.E. in "Computer Engineering"***.

Submitted by:

Bhuvan Shingade	**ID No: TU3S2122006**
Suyash Kasar	**ID No: TU3F2021125**
Ved Kolambkar	**ID No: TU3S21220012**
Yash Matha	**ID No. TU3S21220011**

Examiners Name & Signature:

1.--

2.--

Date: --------------------------------

Place: --------------------------------

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Bhuvan Shingade	**ID No. TU3S2122006**	-------------------
Suyash Kasar	**ID No. TU3F2021125**	-------------------
Ved Kolambkar	**ID No: TU3S2122012**	-------------------
Yash Matha	**ID No. TU3S21220011**	-------------------

Date: ______________________

Place: ______________________

Acknowledgement

We would like to express our sincere gratitude towards our guide **Prof. Rohini Palve**, Major Project Coordinator **Prof. Pooja Singh** for their help, guidance and encouragement, they provided during the project development. This work would have not been possible without their valuable time, patience and motivation. We thank them for making our stint thoroughly pleasant and enriching. It was great learning and an honor being their student.

We are deeply thankful to **Dr. Kishore Sakure (H.O.D Computer Department)** and entire team in the Computer Department. They supported us with scientific guidance, advice and encouragement, they were always helpful and enthusiastic and this inspired us in our work.

We take the privilege to express our sincere thanks to **Dr. L. K. Ragha** our Principal for providing the encouragement and much support throughout our work.

Bhuvan Shingade	**ID No. TU3S2122006**	-------------------
Suyash Kasar	**ID No. TU3F2021125**	-------------------
Ved Kolambkar	**ID No: TU3S2122012**	-------------------
Yash Matha	**ID No. TU3S21220011**	-------------------

Date: ______________________

Place: ______________________

INDEX

Abstract

Text summarization is a critical Natural Language Processing (NLP) task with a wide range of applications, from simplifying complex articles to aiding in content curation and efficient information retrieval. Our text summarizer employs a hybrid approach, combining both extractive and abstractive summarization techniques, making it versatile in handling various types of textual data.

In this summarizer, we utilize cutting-edge NLP models, including transformer-based architectures like BERT and GPT-3, to generate high-quality summaries. The system's workflow begins by preprocessing the input text, tokenizing it, and converting it into a numerical format suitable for model processing. This pre-processing also involves part-of-speech tagging, entity recognition, and sentiment analysis to ensure a comprehensive understanding of the content.

Our approach focuses on extractive summarization, where the most salient sentences from the source text are selected and reorganized to create a coherent summary. This is followed by abstractive summarization, in which the system generates novel sentences to convey the main ideas of the document, enhancing readability and coherence.

List of Figures

List of Tables

List of Abbreviations

Abbreviations	**Definition**
TF-IDF	Term Frequency - Inverse Document Frequency
BERT	Bidirectional Encoder Representations from Transformers
AWS	Amazon Web Services
CI/CD	Continous Integration Continous Deployment

Chapter 1

Introduction

Day by day in this age lots of textual data is increasing. There is need of some way to reduce this data while maintaining the information and meaning of that data. Text Summarization is technique of summarizing the huge small parts of text data into shorter form without changing semantics or meaning of that data. Text summarization is having a great demand in this modern age.

1.1 Introduction to Text Summarizer

The main advantage of Text Summarization is that it reduces the reading time of the user by generating concise and meaningful summary. Text Summarization has classified into two methods. These are Extractive and Abstractive Text summarization approaches. Summarization is the process of automatically generating natural language summaries from an input by maintaining semantics of text. It helps in easy and fast retrieval of information. [1]

Generally, the processing architecture of all the text summarization algorithms have 3 main steps. The first step is Pre-processing step to identify the words, sentences and other components of the text. The second step is processing step which converts the input text to a summary by using a text summarization method. The third is post-processing, which fixes the problem in the created summary.

Majority of the systems supports the extractive based text summarization approach as it copies the text from the original documents/ from the articles. Also, it gives grammatically correct ,relatable and meaningful summary. But they don’t summarize long and complex texts. We mainly focus on extractive text summarization and it’s implementation.

1.2Aim & Objectives:

AIM: The aim of this text summarization project is to develop an efficient and effective text summarizer that can automatically generate high-quality summaries from input texts, enhancing information retrieval and comprehension in various domains.

OBJECTIVES:

1) Ensure the summarization system can adapt to different domains and fields, allowing it to summarize content from a wide range of subjects.

2) Implement techniques for generating abstractive summaries that maintain context, coherence, and the overall essence of the original text.

3) Develop algorithms for the automatic extraction of key information and relevant content from input documents.

1.3 Scope :

Text summarization holds significant promise in an increasingly data-rich world, offering a wide scope of applications. It facilitates the extraction of essential information from lengthy documents, aiding users in swiftly grasping main ideas and key details, with applications spanning news aggregation, research paper scanning, and beyond. Summarizers also enhance information retrieval, enabling search engines and databases to provide users with previews of documents, improving search efficiency and the user experience. Furthermore, text summarization contributes to content generation, generating concise versions of articles, reports, or product descriptions for e-commerce, marketing, and executive summaries. In the realm of language translation, it assists in both translation and summarization, preserving original context and meaning. Legal and medical fields find value in summarization for reviewing contracts, research papers, patient records, and more, expediting decision-making processes. Social media, chatbots, and content curation all benefit from summarization, while individuals can leverage these tools for personal productivity, such as quickly understanding emails and news articles. As technology continues to advance, text summarization's scope is expected to expand, potentially finding applications in education, research, and numerous other domains yet to be explored.[3]

Chapter 2

2.1 Existing Literature Survey

Text summarization, a key component of Natural Language Processing (NLP), involves condensing lengthy texts while retaining essential content. Two primary approaches, extractive and abstractive summarization, utilize techniques like TF-IDF, PageRank, and neural networks. Linguistic features, such as part-of-speech tagging and named entity recognition, aid in entity preservation. Pre-trained models like BERT and GPT, along with evaluation metrics like ROUGE and BLEU, have advanced the field. Recent research has also explored controllable abstractive summarization. Overall, text summarization aims to create concise, contextually relevant summaries to manage information overload efficiently. [2]
This chapter introduces previous work done related to this concept. Some of the research papers and their summary is given below:-

2.2 Literature Survey Table:

Reference no.	Publish year	Research paper	Summary	ROUGE Score
[3]	2020	A Deep Reinforced Model for Extractive Summarization	The paper proposes a sequence-to-sequence (Seq2Seq) model for abstractive summarization. It employs reinforcement learning (RL) techniques to enhance the summarization performance. **Key Findings:** The deep reinforced model significantly improves abstractive summarization, generating more human-like and coherent summaries compared to traditional approaches.	0.345

[6]	2023	Pointer-Generator Networks for Text Summarization	The research introduces pointer-generator networks with attention mechanisms and a copy mechanism. These mechanisms allow the model to copy words directly from the input. **Key Findings:** The pointer-generator networks demonstrate improved performance, especially when dealing with unfamiliar or domain-specific words that are not present in the vocabulary.	0.403
[9]	2022	Get To The Point: Summarization with Pointer-Generator Network	This paper extends the pointer-generator network with a coverage mechanism. The coverage mechanism helps the model to avoid repetition in the generated summaries. **Key Findings:** The proposed extension with the coverage mechanism leads to better summaries, reducing redundancy and producing more informative and non-repetitive results.	0.712
[4]	2021	A Deep Reinforced Model for Abstractive	Proposed a reinforcement learning-based approach for abstractive summarization, addressing the issue	0.267

		Summarization	of generating fluent and informative summaries. **Key Findings:** A reinforcement learning-based approach can lead to more fluent and coherent abstractive summaries compared to traditional methods. Reinforcement learning allows the model to learn and optimize its summaries over time, leading to better quality outputs.	
[7]	2023	Forecasting the Fury: A Deep Learning Approach to Predicting Cyclone Intensity	This research demonstrates the effectiveness of employing ResNet-50 in deep learning for cyclone intensity estimation using INSAT-3D IR imagery. With superior performance metrics such as lower RMSE at 11.69, ResNet-50 emerges as a robust choice, offering a promising solution for precise and timely cyclone intensity assessment crucial for disaster management.	0.489
[8]	2022	Extractive Summarization with BERT(Bidirectional Encoder Representations from Transformers)	Proposed a method to use BERT for extractive, summarization, highlighting the effectiveness of pre-trained embeddings in improving extractive, summarization performance. Key Findings: BERT's ability to understand the context and semantics of text can lead to more accurate identification of relevant sentences or passages in the source document, thereby improving the informativeness of the extractive summary.	0.678

2.3 Problem Statement :

Text Summarization is most demanding area of research for both the retrieving different types of information and for the Natural Language Processing communities. Text Summarization is mostly used in the research articles, in different commercial sector such as in the Telephone communication industry in word Processing tools. Automatic text summarization is an important step for managing different information tasks. It solves the problem of selecting the most important portions of the text from the documents. High quality summarization requires practical Natural Languagae Processing Techniques.

Developing an advanced text summarization system that can automatically generate concise and coherent summaries from lengthy documents while preserving the essential meaning, contextual relevance, and domain-specific information for improved information retrieval, content understanding, and time-saving in various professional and academic contexts.

Chapter 3

3.1 Software model

Model: Scrum[4]

Scrum, an Agile software development framework, can be effectively applied to the development of a text summarizer. Here's how the Scrum methodology can be implemented for this project:

1. Project Initiation:

Define Vision: Clearly outline the vision and primary objectives of the text summarizer project, such as creating an efficient and accurate tool for extracting key information from large volumes of text.

2. Requirements Gathering:

Product Backlog: Create a product backlog that lists all the essential features, functionalities, and data sources required for the text summarizer.

3. Sprint Planning:

Sprint Definition: Organize the project into a series of time-bound sprints, typically lasting two to four weeks, each focusing on specific aspects of the text summarization tool.

Sprint Backlog: For each sprint, select items from the product backlog and define the tasks and requirements necessary to complete them.

4. Development:

Iterative Work: Within each sprint, the development team works iteratively on tasks such as text preprocessing, algorithm development, and user interface design.

Daily Standups: Conduct daily stand-up meetings to monitor progress, discuss obstacles, and make necessary adjustments.

5. Testing and Validation:

Continuous Testing: Testing and validation are integral throughout the project, ensuring the accuracy and effectiveness of the text summarizer.

User Acceptance Testing (UAT): Involve users and stakeholders to provide feedback and validate the tool's results.

6. Review and Adapt:

Sprint Review: At the end of each sprint, hold a review to evaluate progress, showcase results, and gather feedback from team members and stakeholders.

Sprint Retrospective: Conduct a retrospective to identify areas for improvement in the tool's accuracy, user experience, or development processes.

7. Documentation and Knowledge Sharing:

Continuous Documentation: Maintain ongoing documentation, capturing insights, algorithms, and lessons learned throughout the project.

Knowledge Sharing: Foster a collaborative environment by sharing insights and knowledge with the team and stakeholders.

8. Scaling and Integration:

As the text summarizer matures, consider scaling the system to handle larger volumes of text and explore integration opportunities with other applications or platforms.

9. Continuous Improvement:

Embrace a culture of continuous improvement, periodically reassessing the project's objectives and adapting strategies to align with evolving requirements and technological advancements.

10. Deployment and Operational Use:

Deploy the text summarization tool for operational use once it consistently demonstrates accuracy and reliability. Provide training and support for users and system integration.

11. Monitoring and Maintenance:

Continuously monitor the tool's performance and gather feedback from users. Regular maintenance and updates will help maintain the tool's relevance and effectiveness.

Scrum's adaptability, stakeholder involvement, and regular progress assessments make it a suitable framework for developing a text summarizer that can evolve to meet changing requirements and advances in text analysis and NLP technologies.

3.2 Proposed System

The proposed text summarizer system introduces an advanced solution for efficient information extraction from textual content. Leveraging state-of-the-art Natural Language Processing (NLP) and machine learning techniques, it aims to generate precise and coherent summaries. Here are the key components and features of the system:

The system will handle diverse sources of textual data, performing essential tasks such as tokenization, stop-word removal, stemming, and lemmatization. This comprehensive preprocessing ensures that the text is ready for effective summarization.

The system will offer an intuitive user interface, whether web-based or standalone, for easy text input and summary output. It will support batch processing and integration with other applications through APIs. To streamline content summarization, the system will integrate with various data sources, including websites, news feeds, and document repositories, enabling automated and efficient content extraction. [5]

This proposed text summarizer system is poised to deliver a powerful and versatile solution for extracting essential information from text. With its ability to accommodate diverse languages, customization options, and robust evaluation metrics, it will cater to a wide range of domains, from content curation and research to news aggregation and beyond. The system holds the potential to significantly improve the efficiency of information retrieval and processing in various applications.

3.3 SRS

3.3.1. Introduction

3.3.1.1 Purpose

The purpose of this document is to provide a detailed outline of the functional and non-functional requirements for the development of a sophisticated Text Summarizer application that utilizes cutting-edge Natural Language Processing (NLP) techniques.

3.3.1.2 Scope

The Text Summarizer will be a user-friendly, web-based application catering to professionals, students, and researchers who frequently deal with lengthy documents. The application will allow users to input academic papers, articles, or any text containing substantial information and receive well-structured summaries that capture the core concepts and essential details.

3.3.2. System Requirements

3.3.2.1 Functional Requirements

3.3.2.1.1 Text Input

The system shall present a clean and intuitive user interface that facilitates the effortless input of textual content for summarization.

The input text field shall accommodate documents of varying lengths, ranging from a minimum of 500 words to several thousand words.

3.3.2.1.2 Text Summarization

The system shall employ advanced NLP techniques, including word embeddings and attention mechanisms, to deeply understand the input text.

The summarizer shall employ abstractive techniques, rephrasing and rewording sentences when necessary, to produce coherent and contextually appropriate summaries.

Users shall have the flexibility to choose between different summarization options, such as "key points only," "key points with context," and "comprehensive summary."

Summary Output

The system shall render the generated summary within the user interface, highlighting important sentences and concepts.

Users shall be able to adjust the length of the summary dynamically by specifying the desired number of sentences or words.

3.3.2.1 Non-Functional Requirements

3.3.2.1 Performance

The summarization process shall exhibit optimal performance, delivering summaries within 3 seconds for input texts ranging up to 2000 words.

The system's architecture shall be designed to accommodate spikes in usage, ensuring consistent performance even during peak usage hours.

3.3.2.2 Accuracy

The summarization algorithm shall be trained on a diverse range of text genres, including scientific articles, news articles, and technical documents, to ensure accuracy across different contexts.

The system shall aim for a ROUGE score of at least 0.5, reflecting a high level of concordance between the generated summaries and human-written references.

3.3.2.3 User Interface

The user interface shall adhere to modern design principles, supporting both desktop and mobile devices.

the most of the summarization features.

3.3.2.4 Security and Privacy

The application shall employ end-to-end encryption to safeguard user-provided text data during transmission and storage.

The backend servers shall implement stringent access controls and authentication mechanisms to prevent unauthorized access to sensitive user data.

3.3.3 System Architecture

3.3.3.1 Components

User Interface (UI): Facilitates user interaction, allowing input of texts and displaying generated summaries.

NLP Processing Module: Employs pre-trained language models and embeddings to comprehend the textual content.

Summarization Algorithm: Leverages advanced abstractive techniques to craft accurate and coherent summaries.

Backend Server: Orchestrates communication between the UI, NLP Processing Module, and Summarization Algorithm.

3.3.3.2 Interaction

The UI shall capture user input, and upon submission, the backend server shall route the input text to the NLP Processing Module.

The NLP Processing Module shall apply attention mechanisms and linguistic analysis to determine the text's salient content.

The Summarization Algorithm shall create a well-structured summary, which is relayed back to the UI through the backend server for presentation to the user.

3.3.4 Testing Requirements

3.3.4.1 Unit Testing

The NLP Processing Module shall undergo rigorous unit testing, evaluating its proficiency in identifying key sentences across diverse text samples.

The Summarization Algorithm shall be subjected to unit tests using both generic and domain-specific texts to ensure consistent quality in summarization.

3.3.4.2 Integration Testing

Integration tests shall validate the seamless interaction between the UI, Backend Server, NLP Processing Module, and Summarization Algorithm.

Mock data sets representative of real-world scenarios shall be used to ensure stability and reliability.

3.3.4.3 User Acceptance Testing

A diverse group of potential users shall participate in comprehensive user acceptance testing, evaluating the accuracy, usability, and overall experience of the application.

Feedback gathered from this testing phase shall be used to fine-tune and enhance the application's performance and user interface.

3.3.5. Conclusion

This comprehensive Software Requirements Specification defines the intricate details of a sophisticated Text Summarizer using NLP techniques. By adhering to these requirements, the development process will yield an innovative application that empowers users to effortlessly distill

lengthy textual content into concise, meaningful summaries. The application's functionality, accuracy, and user-centric design will undoubtedly make it a valuable tool for professionals and academics alike.

3.4 Hardware and Software Requirements

3.4.1 Hardware Requirement

SR.NO	NAME OF THE COMPONENTS	SPECIFICATION
1.	OPERATING SYSTEM	WINDOWS 10
2.	PROCESSOR	Ryzen 5
3.	RAM	8GB
4.	HARD DISK	1 TB

3.4.2 Software Requirement

SR.NO	NAME OF THE COMPONENT	SPECIFICATION
1.	LANGUAGE	Python
2.	SOFTWARE	Tensorflow, Pytorch
3.	LIBRARIES	Nltk ,Sci-Kit

Chapter 4

Design

4.1 Use case Diagram:

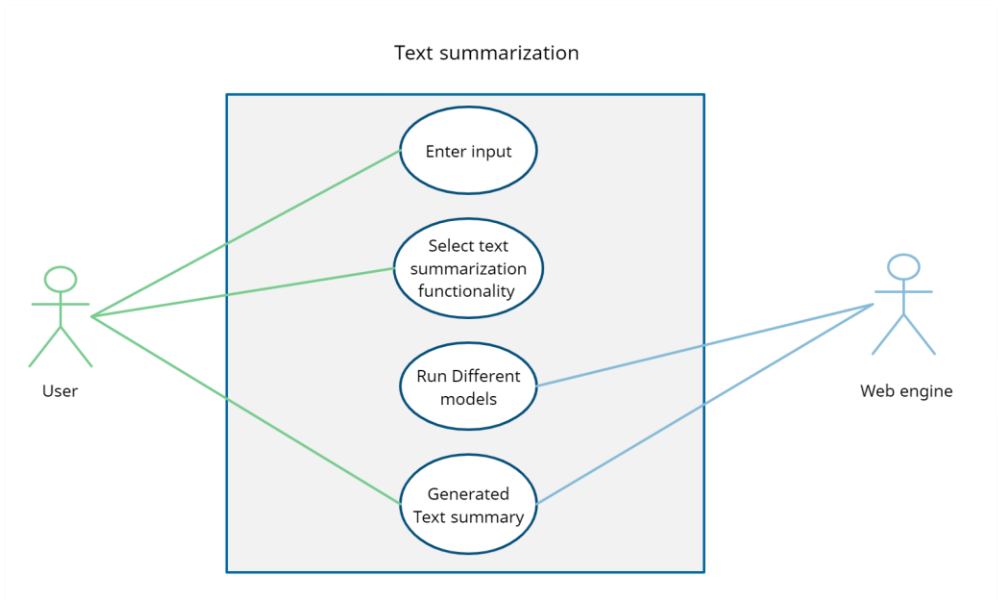


Fig 4.1. Use case diagram

4.2 Sequence Diagram

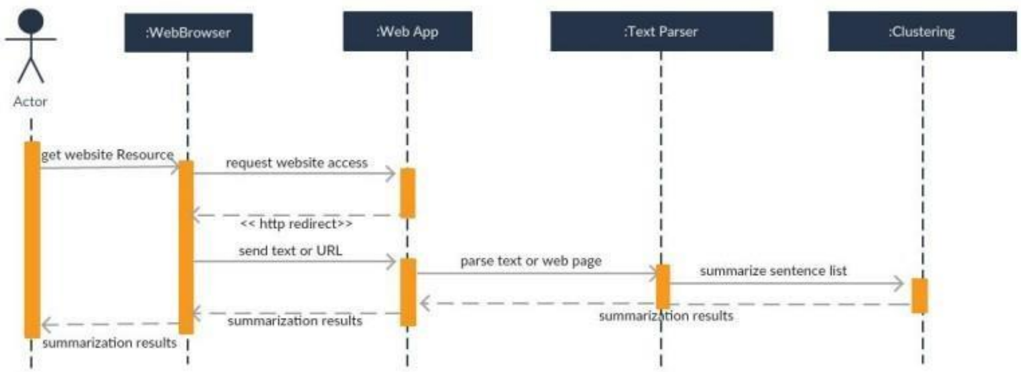


Fig 4.2. Sequence Diagram

Chapter 5

Algorithm

5.1 Algorithms to be used:

5.1.1: Text Rank:

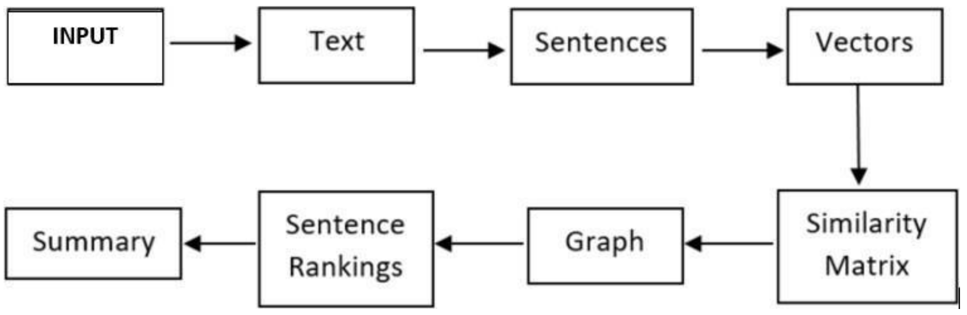


Fig 5.2 Text Rank

A type of graph-based ranking algorithm used for recommendations is called Text Rank. Several applications that employ text phrases use Text Rank. It utilized the information found in the full text to recursively compute the ranking of text phrases. [2]

The text-related graph is created using Text Rank. Each phrase represents a vertex in a network, and each vertex is connected to every other vertex. These vertices voted in favor of that other vertices. The greater number of votes determines each vertex's significance. Such significance is the objective of this ranking is to order the sentences.

The process that Text Rank uses is as follows:

1. Turn documents into sentences using tokens.
2. Clean up every sentence in the text.
3. Count the important words.
4. Calculate the Jaccard distance between sentences and key phrases.
5. Rank the sentences with higher significance. Text Rank is a graph-based algorithm, easy to understand and implement. Its results are less semantic.

Chapter 6

6.1 Methodology:

Experimental Setup

- Gather textual data from various sources, including documents, articles, or web content, to create a diverse dataset for summarization.[8]

- Perform essential preprocessing tasks, such as text tokenization, removal of stop words, punctuation, and special characters, and stemming or lemmatization to standardize and clean the text.

- Identify and select relevant features for text summarization, which may include keyword frequency, sentence importance, and thematic content.

- Implement a combination of extractive and abstractive summarization techniques to provide options for users.
- Extractive: Employ methods like TextRank or LexRank to identify and select important sentences or phrases based on their relevance to the overall content.
- Abstractive: Utilize advanced NLP techniques, including Transformer-based models like BERT or GPT-3, for generating coherent and human-like summaries.

- Offer users the ability to customize summarization parameters, such as desired summary length, level of abstraction, or specific domain focus.
- Fine-tuning options to adapt the summarization model to specific industries or use cases.

- Implement automated evaluation metrics like ROUGE (Recall-Oriented Understudy for Gisting Evaluation), BLEU (Bilingual Evaluation Understudy), and METEOR to assess the quality of generated summaries and provide users with feedback on summarization accuracy.[1][3]

- Develop a user-friendly interface, which can be web-based or standalone, for easy text input and summary output.
- Include options for batch processing and integration with external applications through APIs.
- Enable real-time summarization capabilities, allowing users to receive instantaneous summaries for dynamic and live content streams.
- Deploy the summarization system for use by organizations or individuals, allowing users to upload text for summarization.
- Provide users with succinct, coherent summaries based on their specific preferences and requirements.

This comprehensive methodology encompasses data processing, diverse summarization techniques, customization options, real-time capabilities, and thorough testing to deliver a versatile and effective text summarization system. It caters to a wide range of applications, from content curation and research to news aggregation, enhancing the efficiency of information extraction and processing. [4]

How we Developed our model:

- Text summarization via NLP involves several key steps. First, the text is tokenized into individual words or phrases. Then, a frequency matrix is created to track the occurrence of each word in every sentence. Term Frequency (TF) is calculated for each word in each sentence, indicating its importance within that sentence. Additionally, a table is constructed to represent the presence of words in each sentence, laying the groundwork for Inverse Document Frequency (IDF) calculations.
- IDF is then computed to assess the significance of words across the entire document collection. By combining TF and IDF, the TF-IDF matrix is generated, highlighting words that are important within specific sentences but relatively rare across the corpus. Sentences are scored based on their TF-IDF values, determining their relevance. A threshold score is established to select sentences for inclusion in the summary, ensuring that only the most pertinent information is retained
- Ultimately, the selected sentences form the summary, offering a concise representation of the original text's main points. This process enables efficient

comprehension and extraction of key information from lengthy documents.

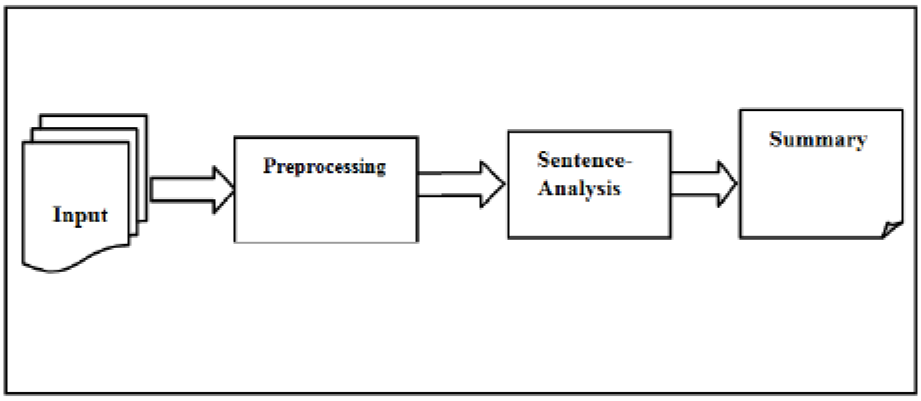


Fig 6.1. General Steps of summarization

1. Tokenize the sentences:

- Tokenization is the process of breaking down a text into individual words, phrases, or symbols, which are known as tokens.
- In natural language processing (NLP), tokenization is often the first step in processing textual data.
- It involves splitting the text into smaller units, such as words or subwords, using predefined rules or patterns.
- For example, the sentence "Hello, how are you?" might be tokenized into ["Hello", ",", "how", "are", "you", "?"].

2. Create the Frequency matrix of the words in each sentence:

- Once the text is tokenized into sentences, the next step is to create a matrix representing the frequency of each word in each sentence.
- Each row of the matrix corresponds to a sentence, and each column corresponds to a unique word in the text.
- The value in each cell represents the frequency of the word in the corresponding sentence.
- For example, if the word "apple" appears twice in the first sentence and once in the second sentence, the matrix might look like this:

	apple	banana	orange
Sentence 1	2	0	1
Sentence 2	1	0	0

3. Calculate Term Frequency (TF) and generate a matrix:

- Term Frequency (TF) is a measure of how often a term (word) appears in a document (sentence), relative to the total number of terms in that document.
- TF is calculated for each word in each sentence by dividing the frequency of the word by the total number of words in the sentence.
- This results in a matrix where each cell represents the TF value of a word in a sentence.
- For example, if a sentence contains 10 words and the word "apple" appears 3 times in that sentence, the TF value for "apple" in that sentence would be 0.3.

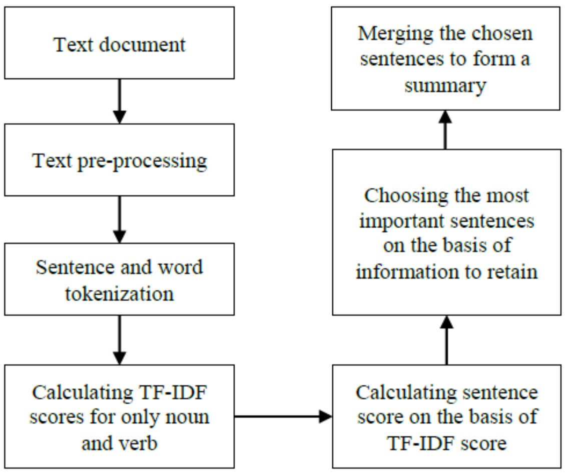

Text document
Text pre-processing
Sentence and word tokenization
Calculating TF-IDF scores for only noun and verb
Calculating sentence score on the basis of TF-IDF score
Choosing the most important sentences on the basis of information to retain
Merging the chosen sentences to form a summary

Fig 6.2. TF-IDF Workflow

4. Creating a table for documents per words:

- This step involves creating a table or matrix where rows represent words and columns represent documents (sentences).
- Each cell in the table indicates whether a particular word appears in a specific document.
- Typically, this table contains binary values: 1 if the word is present in the document and 0 if it is not.
- This table is used to calculate the Inverse Document Frequency (IDF) in the next step.

5. Calculate Inverse Document Frequency (IDF) and generate a matrix:

- Inverse Document Frequency (IDF) is a measure of how much information a word provides across a collection of documents.
- IDF is calculated for each word by taking the logarithm of the total number of documents divided by the number of documents containing the word.

- This results in a matrix where each cell represents the IDF value of a word.
- Words that appear frequently across documents have lower IDF values, while words that appear rarely have higher IDF values.

6. Calculate TF-IDF and generate a matrix:

- TF-IDF (Term Frequency-Inverse Document Frequency) is a statistical measure used to evaluate the importance of a word in a document relative to a collection of documents.
- TF-IDF is calculated by multiplying the TF matrix by the IDF matrix element-wise.
- The resulting matrix represents the importance of each word in each sentence relative to the entire corpus.
- Words with high TF-IDF values are considered important in the context of a specific sentence but are relatively rare across the entire corpus.

7. Score the sentences:

- After obtaining the TF-IDF matrix, the next step is to score each sentence based on the TF-IDF values of the words it contains.
- This can be done by summing the TF-IDF values of all the words in each sentence.
- Sentences with higher scores are considered more important or relevant to the overall content of the text.

8. Find the threshold:

- To determine which sentences to include in the summary, a threshold score needs to be established.
- This threshold can be set based on statistical measures (e.g., mean, median, standard deviation of the sentence scores) or adjusted manually.
- Sentences with scores above the threshold are considered for inclusion in the summary.

- 9. Generate the summary:
- Finally, sentences whose scores exceed the threshold are selected to form the summary.
- These sentences represent the most important or relevant information extracted from

the original text.

- The summary aims to provide a concise overview of the main points covered in the text, making it easier for readers to grasp the key ideas without having to read the entire document.

Deployment:

AWS CI/CD deployment with GitHub Actions revolutionizes software development by seamlessly integrating automated build, test, and deployment processes. Leveraging GitHub Actions' workflow automation capabilities alongside AWS services like EC2, ECS, and Lambda, developers can orchestrate complex deployment pipelines with ease. Continuous integration ensures that code changes are promptly validated and integrated into the codebase, while continuous deployment automates the rollout of these changes to production environments.
By harnessing the power of cloud computing and version control, teams can accelerate the delivery of high-quality software, enhance collaboration, and drive innovation. This modern approach to deployment promotes agility, reliability, and scalability, empowering organizations to deliver value to their customers faster and more efficiently.

1. Login to AWS console:
   - When you log in to the AWS Management Console, you're accessing a web-based interface that allows you to manage various AWS services. This console provides a centralized location to access, configure, and monitor your AWS resources.
   - It's essential to have an AWS account to log in. If you don't have one, you can sign up for an AWS account on the AWS website.

2. Create IAM user for deployment:
   - IAM (Identity and Access Management) allows you to manage access to AWS services and resources securely.
   - When creating an IAM user for deployment, you're creating a dedicated user account specifically for deployment tasks, separate from your root account.
   - You can specify the access permissions for this user by attaching IAM policies. These policies define what actions the user can perform on which AWS resources.
   - It's recommended to follow the principle of least privilege, granting only the permissions necessary for the deployment tasks.

3. Create ECR repo to store/save Docker image:
   - Amazon ECR (Elastic Container Registry) is a fully-managed Docker container registry service provided by AWS.
   - Creating a repository in ECR allows you to store, manage, and deploy Docker container images securely.
   - Each repository in ECR has its own unique URI that can be used to push and pull Docker images.

- You can configure repository settings such as image scanning for vulnerabilities, encryption of images at rest, and immutability of image tags to enhance security and compliance.

4. Create EC2 machine (Ubuntu):
    - Amazon EC2 (Elastic Compute Cloud) provides resizable compute capacity in the cloud, allowing you to launch virtual servers known as instances.
    - When creating an EC2 instance, you choose an Amazon Machine Image (AMI) that serves as the template for the instance's operating system and software.
    - Ubuntu is a popular Linux distribution widely used for its ease of use and extensive package repositories.
    - Selecting the appropriate instance type depends on factors such as CPU, memory, storage, and networking requirements for your workload.

5. Open EC2 and Install Docker in EC2 Machine:
    - Once the EC2 instance is launched, you connect to it using SSH (Secure Shell) to access the command-line interface.
    - Updating the package repository ensures that you have the latest package information available.
    - Installing Docker on the EC2 instance allows you to run and manage Docker containers, which are lightweight, portable, and scalable units of software.
    - Starting and enabling the Docker service ensure that Docker starts automatically upon system boot.

6. Configure EC2 as self-hosted runner:
    - GitHub Actions allows you to automate workflows for your GitHub repositories, including CI/CD pipelines.
    - Self-hosted runners are machines (such as EC2 instances) dedicated to running GitHub Actions workflows in your own environment.
    - Configuring an EC2 instance as a self-hosted runner involves downloading and configuring the GitHub Actions runner software on the instance.
    - This runner software communicates with GitHub to execute workflows triggered by events in your repository.

7. Setup GitHub secrets:
    - GitHub Secrets are encrypted environment variables that you can use in your GitHub Actions workflows.
    - Storing sensitive information such as AWS credentials as secrets ensures that they are not exposed in your repository.

- These secrets can be accessed by your workflows securely, allowing you to authenticate with AWS services and perform deployment tasks without compromising security.

By thoroughly understanding and following these detailed steps, you can establish a robust CI/CD deployment pipeline using AWS services and GitHub Actions, ensuring efficient and secure deployment of your applications.

Chapter 7
Performance Evaluation

7.1 Rouge Metric:

ROUGE (Recall-Oriented Understudy for Gisting Evaluation), is a set of metrics and a software package specifically designed for evaluating automatic summarization, but that can be also used for machine translation. The metrics compare an automatically produced summary or translation against reference (high-quality and human-produced) summaries or translations.

7.1.1 ROUGE-N

ROUGE-N measures the number of matching n-grams between the model-generated text and a human-produced reference.

Consider the reference R and the candidate summary C:

R: The cat is on the mat.

C: The cat and the dog.

ROUGE-1

Using R and C, we are going to compute the precision, recall, and F1-score of the matching n-grams. Let’s start computing ROUGE-1 by considering 1-grams only.

ROUGE-1 precision can be computed as the ratio of the number of unigrams in C that appear also in R (that are the words “the”, “cat”, and “the”), over the number of unigrams in C.

ROUGE-1 precision = 3/5 = 0.6

ROUGE-1 recall can be computed as the ratio of the number of unigrams in R that appear also in C (that are the words “the”, “cat”, and “the”), over the number of unigrams in R.

ROUGE-1 recall = 3/6 = 0.5

Then, ROUGE-1 F1-score can be directly obtained from the ROUGE-1 precision and recall using the standard F1-score formula.

ROUGE-1 F1-score = 2 * (precision * recall) / (precision + recall) = 0.54

ROUGE-2

Let’s try computing the ROUGE-2 considering 2-grams.

Remember our reference R and candidate summary C:

R: The cat is on the mat.

C: The cat and the dog.
ROUGE-2 precision is the ratio of the number of 2-grams in C that appear also in R (only the 2-gram “the cat”), over the number of 2-grams in C.
ROUGE-2 precision = 1/4 = 0.25
ROUGE-1 recall is the ratio of the number of 2-grams in R that appear also in C (only the 2-gram “the cat”), over the number of 2-grams in R.
ROUGE-2 recall = 1/5 = 0.20
Therefore, the F1-score is:

ROUGE-2 F1-score = 2 * (precision * recall) / (precision + recall) = 0.22

7.1.2 ROUGE-L:

ROUGE-L is based on the longest common subsequence (LCS) between our model output and reference, i.e. the longest sequence of words (not necessarily consecutive, but still in order) that is shared between both. A longer shared sequence should indicate more similarity between the two sequences.

We can compute ROUGE-L recall, precision, and F1-score just like we did with ROUGE-N, but this time we replace each n-gram match with the LCS.

Remember our reference R and candidate summary C:
R: The cat is on the mat.
C: The cat and the dog.

The LCS is the 3-gram “the cat the” (remember that the words are not necessarily consecutive), which appears in both R and C.
ROUGE-L precision is the ratio of the length of the LCS, over the number of unigrams in C.
ROUGE-L precision = 3/5 = 0.6
ROUGE-L precision is the ratio of the length of the LCS, over the number of unigrams in R.
ROUGE-L recall = 3/6 = 0.5
Therefore, the F1-score is:

ROUGE-L F1-score = 2 * (precision * recall) / (precision + recall) = 0.55

7.1.3 ROUGE-S:

ROUGE-S allows us to add a degree of leniency to the n-gram matching performed with ROUGE-N and ROUGE-L. ROUGE-S is a skip-gram concurrence metric: this allows to search for consecutive words from the reference text that appear in the model output but are separated by one-or-more other words.

Consider the new reference R and candidate summary C:

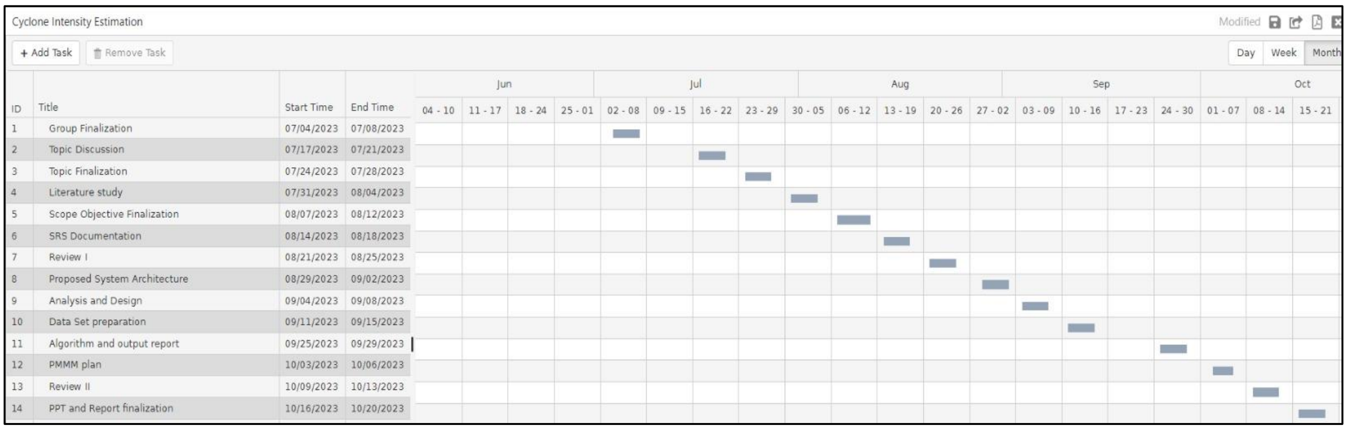
R: The cat is on the mat.
C: The gray cat and the dog.
If we consider the 2-gram “the cat”, the ROUGE-2 metric would match it only if it appears in C exactly, but this is not the case since C contains “the gray cat”. However, using ROUGE-S with unigram skipping, “the cat” would match “the gray cat” too.

We can compute ROUGE-S precision, recall, and F1-score in the same way as the other ROUGE metrics.

Chapter 8

Problem Timeline

8.1 Gantt chart:



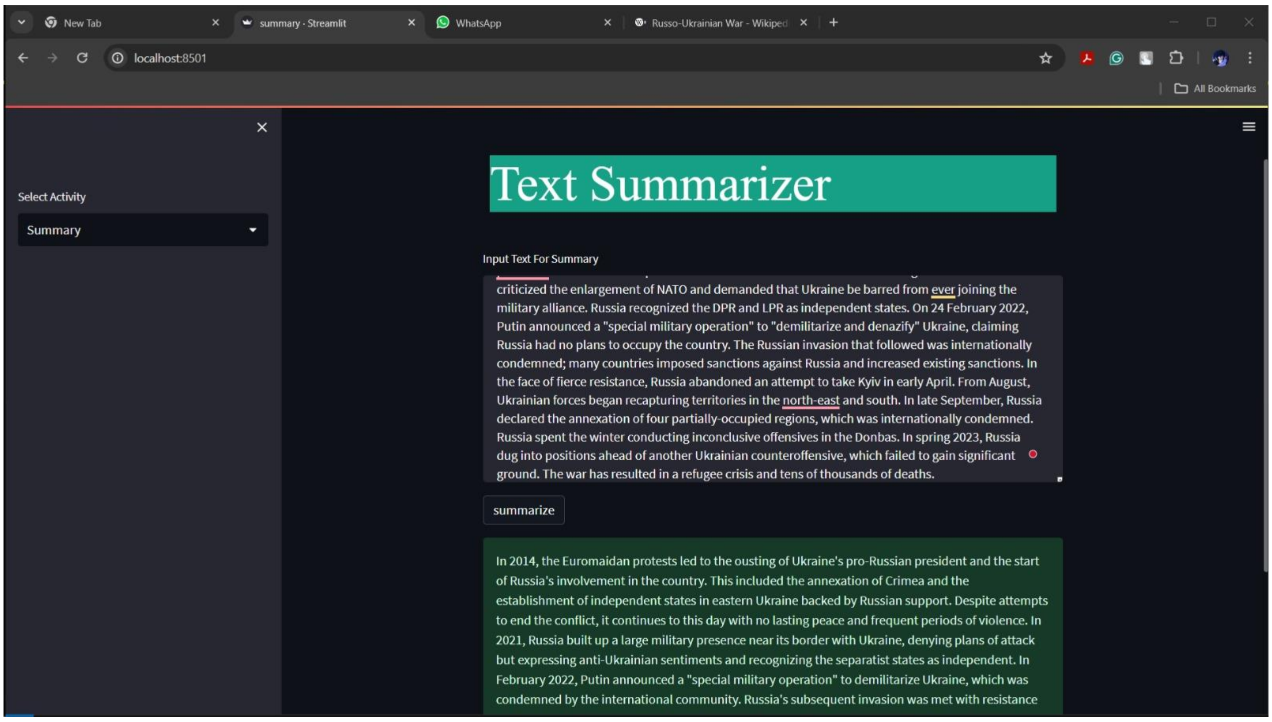
Cyclone Intensity Estimation

ID	Title	Start Time	End Time
1	Group Finalization	07/04/2023	07/08/2023
2	Topic Discussion	07/17/2023	07/21/2023
3	Topic Finalization	07/24/2023	07/28/2023
4	Literature study	07/31/2023	08/04/2023
5	Scope Objective Finalization	08/07/2023	08/12/2023
6	SRS Documentation	08/14/2023	08/18/2023
7	Review I	08/21/2023	08/25/2023
8	Proposed System Architecture	08/29/2023	09/02/2023
9	Analysis and Design	09/04/2023	09/08/2023
10	Data Set preparation	09/11/2023	09/15/2023
11	Algorithm and output report	09/25/2023	09/29/2023
12	PMMM plan	10/03/2023	10/06/2023
13	Review II	10/09/2023	10/13/2023
14	PPT and Report finalization	10/16/2023	10/20/2023

Fig 8.1 Gantt Chart

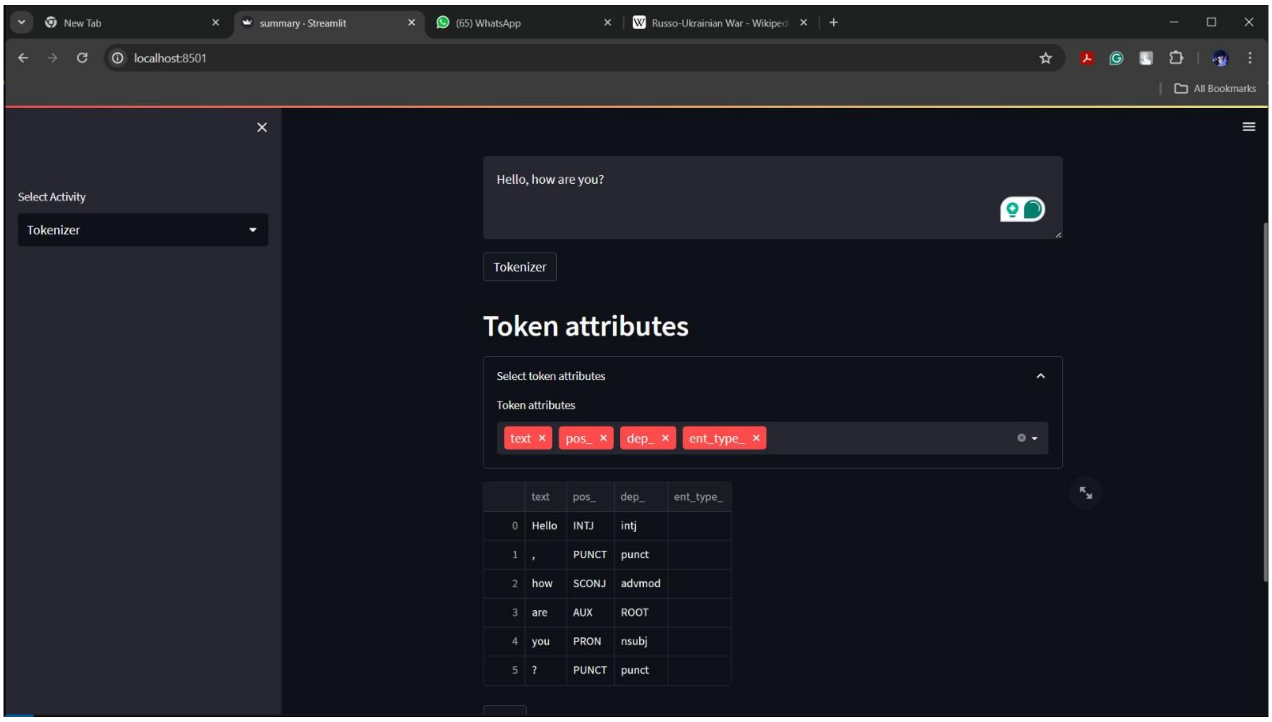
Chapter 9

Results



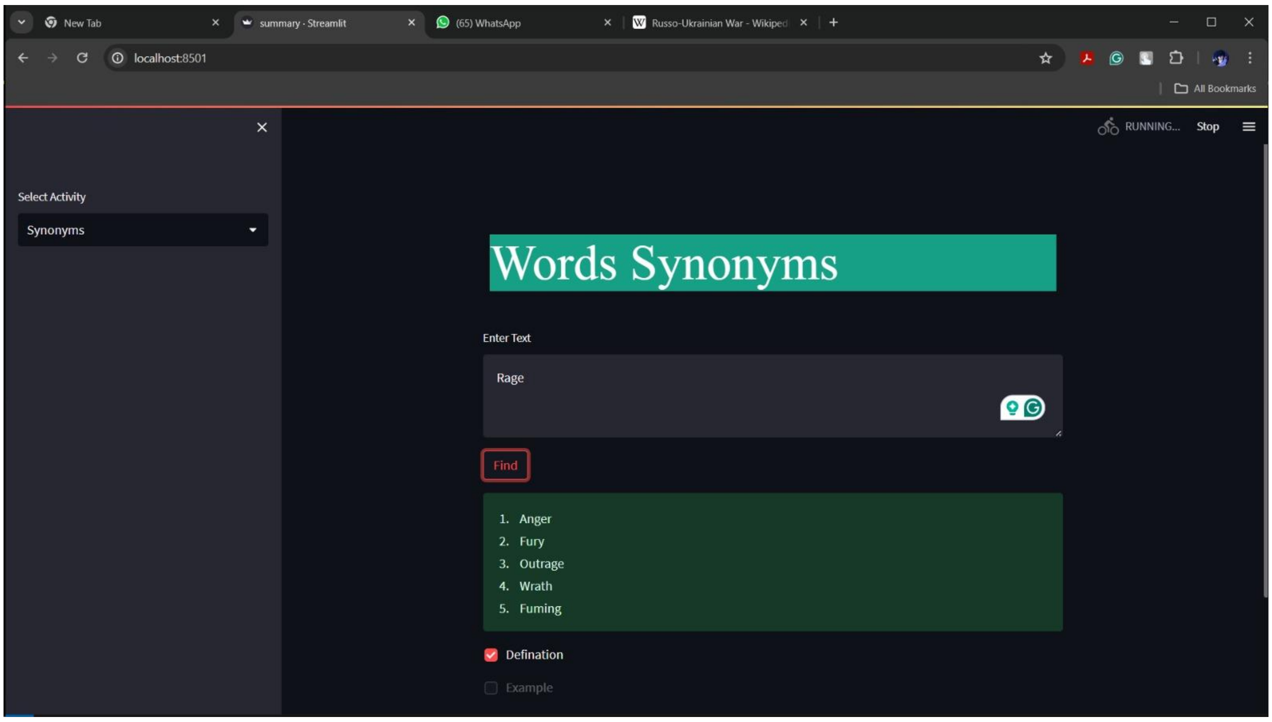
criticized the enlargement of NATO and demanded that Ukraine be barred from ever joining the military alliance. Russia recognized the DPR and LPR as independent states. On 24 February 2022, Putin announced a "special military operation" to "demilitarize and denazify" Ukraine, claiming Russia had no plans to occupy the country. The Russian invasion that followed was internationally condemned; many countries imposed sanctions against Russia and increased existing sanctions. In the face of fierce resistance, Russia abandoned an attempt to take Kyiv in early April. From August, Ukrainian forces began recapturing territories in the north-east and south. In late September, Russia declared the annexation of four partially-occupied regions, which was internationally condemned. Russia spent the winter conducting inconclusive offensives in the Donbas. In spring 2023, Russia dug into positions ahead of another Ukrainian counteroffensive, which failed to gain significant ground. The war has resulted in a refugee crisis and tens of thousands of deaths.

In 2014, the Euromaidan protests led to the ousting of Ukraine's pro-Russian president and the start of Russia's involvement in the country. This included the annexation of Crimea and the establishment of independent states in eastern Ukraine backed by Russian support. Despite attempts to end the conflict, it continues to this day with no lasting peace and frequent periods of violence. In 2021, Russia built up a large military presence near its border with Ukraine, denying plans of attack but expressing anti-Ukrainian sentiments and recognizing the separatist states as independent. In February 2022, Putin announced a "special military operation" to demilitarize Ukraine, which was condemned by the international community. Russia's subsequent invasion was met with resistance



Hello, how are you?

	text	pos_	dep_	ent_type_
0	Hello	INTJ	intj	
1	,	PUNCT	punct	
2	how	SCONJ	advmod	
3	are	AUX	ROOT	
4	you	PRON	nsubj	
5	?	PUNCT	punct	



Select Activity

Synonyms

Words Synonyms

Enter Text

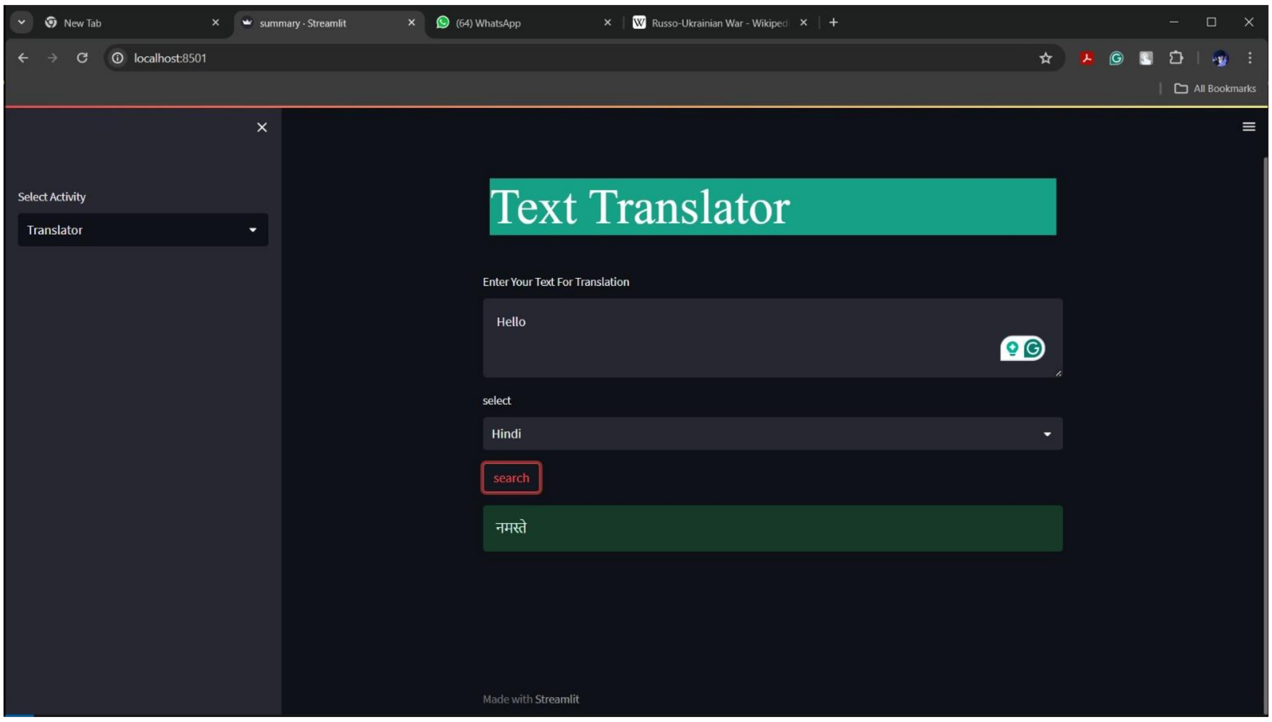
Rage

Find

1. Anger
2. Fury
3. Outrage
4. Wrath
5. Fuming

[✓] Defination

[ ] Example



Select Activity

Translator

Text Translator

Enter Your Text For Translation

Hello

select

Hindi

search

नमस्ते

Chapter 10

Conclusion:

Although text summarization is an old challenge, current research is focused on emerging trends in biomedicine, product reviews, education do- mains, emails, and blogs. In these areas, especially on the World Wide Web, there is an overload of information. In the field of NLP (Natural Language Processing), automatic summarization is an active area of research. It in-volves automatically creating a summary from a text or series of texts. Using extractive document summarization, you can automatically select several indicative sentences, passages, or paragraphs from the original document Some approaches to text summarization, such as Neural Networks, Graph Theory, Fuzzy, and Cluster, have been successful in making an effective summary. Both extractive and abstractive methods have been researched. Most summarization techniques are based on extractive methods.

References

[1] Kanithi Purna Chandu “Text Summarization Using Natural Language Processing”, International Journal of Research Publication and Reviews, Vol 3, no 11, pp 649-655,.
[2] Elena Lloret, Manual Palomar “Text summarisation in progress: A literature review”, Research Gate
[3] Dr. Rashmi Sharma, Shivam Chaudhary, Sejal Tyagi “TEXT SUMMARIZER USING NLP NATURAL LANGUAGE PROCESSING”, International Research Journal of Modernization in Engineering Technology and Science
[4] Reeta Rani, Sawal Tandon, “LITERATURE REVIEW ON AUTOMATIC TEXT SUMMARIZATION”, 2019.
[5] Sheetal Patil, Avinash Pawar, Siddhi Khanna, Anurag Tiwari, Somay Trivedi “Text Summarizer Using NLP”, 2022.
[6] Aakash Srivastava, Kamal Chauhan, Himanshu Daharwal, Nikhil Mukati, Pranoti Shrikant Kavimandan “Text Summarizer Using NLP (Natural Language Processing)” , 2020.
[7] Divakar Yadav, Jalpa Desai, Arun Kumar Yadav, “Automatic Text Summarization Methods: A Comprehensive Review” , 2023.
[8] ChetanaVaragantham, J.SrinijaReddy, UdayYelleni, MadhumithaKotha, Dr P.VenkateswaraRao, “TEXT SUMMARIZATION USING NLP”, International Conference on Industry 4.0 Technology (I4Tech), 2022.
[9] Adhika Pramita Widyassari, Supriada Rustad, Guruh Fajar Shidik, Edi Noersasongko, Abdul Syukur, Affandy Affandy, De Rosal Ignatius Moses Setiadi “Review of automatic text summarization techniques & methods” , Journal of King Saud University, 2020.
[10] Ravali Boorugu, Dr. Gajula Ramesh, Dr. Karanam Madhavi, “Summarizing Product Reviews Using NLP Based Text Summarization”, International Journal of Scientific & Technology Research Vol 8, Issue 10, 2019
[11] Waseemullah, Zainab Fatima Shehnila Zardari, Muhammad Fahim, Maria Andleeb Siddiqui, Ag. Asri Ag. Ibrahim, Kashif Nisar, Laviza Falak Naz, “A Novel Approach for Semantic Extractive Text Summarization”, MDPI, 2022.
[12] Hamza Shabbir Moiyadi, Harsh Desai, Dhairya Pawar, Geet Agrawal, Nilesh M.Patil, “NLP Based Text Summarization Using Semantic Analysis”, 2016
[13] Ha Nguyen Thi Thu, “An Optimization Text Summarization Method Based on Na¨ıve Bayes and Topic Word for Single Syllable Language”,Applied Mathematical Sciences, Vol. 8, 2020 no. 3, 99 – 115, HIKARI Ltd, www.m-ikari.com
[14] International Arab Journal of E-Technology, 1(4). Bird, S., Klein, E., Loper, E. (2019) Natural language processing with Python. United States: O’Reilly Media), 164- 168.
[15] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint arXiv:1810.04805.
[16] Nallapati, R., Zhai, F., & Zhou, B. (2016). Abstractive Text Summarization using Sequence-to-Sequence RNNs and Beyond. arXiv preprint arXiv:1602.06023.
[17] Kulkarni, A. R., Apte, S. S. (2013). A domain-specific automatic text summarization using Fuzzy Logic. International Journal of Computer Engineering and Technology (IJCET), 4(4), 449-461.
[18] Das, D., Martins, A. F. (2021). A survey on automatic text summarization. Literature Survey for the Language and Statistics, 3(3), 1-12
[19] Li, P., & Lam, W. (2020). Text Summarization Techniques: A Comprehensive Survey. Journal of Artificial Intelligence Research, 68, 299-356.
[20] Chen, J., Li, T., & Zhou, L. (2019). A Survey on Neural Network-based Summarization Methods. Information, 10(8), 251.
[21] Rupal Bhargava.et al. (2022), “ATSSI: Abstractive Text Summarization using Sentiment Infusion” Twelfth International Multi-Conference on Information Processing.