A

MINOR PROJECT REPORT

On

TEXT SUMMARIZATION USING NLP

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

B. TECH

in

COMPUTER SCIENCE AND ENGINEERING

Submitted By

Kishore Ramesh	:	**RA2011026040040**
Mohamed Yaseen MS	:	**RA2011026040020**
Saguturu Kishan Sai	:	**RA2011026040031**

DEPT. OF COMPUTER SCIENCE & ENGINEERING
SRM Institute of Science & Technology
Vadapalani Campus, Chennai

OCTOBER 2023

A

MINOR PROJECT REPORT

On

TEXT SUMMARIZATION USING NLP

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

B. TECH
in
COMPUTER SCIENCE AND ENGINEERING

Submitted By

Kishore Ramesh	:	**RA2011026040040**
Mohamed Yaseen MS	:	**RA2011026040020**
Saguturu Kishan Sai	:	**RA2011026040031**

DEPT. OF COMPUTER SCIENCE & ENGINEERING
SRM Institute of Science & Technology
Vadapalani Campus, Chennai

OCTOBER 2023

BONAFIDE CERTIFICATE

Certified that this project report **“Text Summarization using NLP”** is the Bonafide work of **Kishore Ramesh, Mohamed Yaseen MS and Saguturu Kishan Sai** having Registration Number **RA2011026040040, RA2011026040020 and RA2011026040031** respectively, who carried out the project work under my supervision.

SIGNATURE OF THE GUIDE	**SIGNATURE OF THE HOD**
Mr. Muthurasu N	Dr. S. Prasanna Devi, B.E., M.E., Ph.D., PGDHRM., PDF(IISc)
Assistant Professor	Professor
Department of Computer Science and Engineering	Department of Computer Science and Engineering
SRM Institute of Science & Technology	SRM Institute of Science & Technology
Vadapalani Campus	Vadapalani Campus

ACKNOWLEDGEMENT

We express our humble gratitude to **Dr.C.Muthamizhchelvan**, Vice Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support. We extend our sincere thanks to our Dean, **Dr.C.V.Jayakumar**, SRM Institute of Science and Technology, Vadapalani campus for his invaluable support. We wish to thank **Dr.S.Prasanna Devi**, Professor & Head, Department of CSE, SRM Institute of Science and Technology, Vadapalani Campus for her valuable suggestions and encouragement throughout the period of the project work.

We are extremely grateful to our Project Coordinator, Ms. S. Niveditha, Department of CSE, SRM Institute of Science and Technology, Vadapalani Campus, for leading and helping us to complete our course.

Our inexpressible respect and thanks to my guide, **Mr. Muthurasu N**, **Assistant Professor**, Department of CSE, SRM Institute of Science and Technology, Vadapalani Campus for providing me an opportunity to pursue my project under his/her mentorship.

We sincerely thank our Management, staff and students of the Department of CSE, SRM Institute of Science and Technology, Vadapalani Campus who have directly or indirectly helped our project. Finally, we would like to thank my parents, our family members and our friends for their unconditional love, constant support and encouragement

Kishore Ramesh
Mohamed Yaseen M S
Saguturu Kishan Sai

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

The field of natural language processing (NLP) plays a crucial role in text summarization, which involves condensing large texts into concise summaries while preserving the essential information. This abstract explores various strategies employed in text summarization, including extractive and abstractive methods. Extractive methods involve selecting significant sentences or phrases from the original text, while abstractive methods generate new sentences that capture the main ideas. However, both approaches face challenges such as maintaining coherence, using domain-specific language, and avoiding information loss.

Thanks to advancements in machine learning and deep learning, significant progress has been made in developing high-quality summarization models. These improvements have enabled the creation of effective summaries across different topics and languages. In this project, the Python libraries Spacy and Streamlit, along with the NewsAPI, are utilized to implement text summarization.

Text summarization finds applications in various domains, including news aggregation, document summarization, and information retrieval. By providing concise summaries, this technology assists individuals in making informed decisions and consuming information more efficiently. News aggregation platforms can utilize text summarization to present users with key points from multiple news articles, saving them time and effort. Document summarization aids in quickly understanding the main ideas of lengthy documents, enabling users to extract relevant information efficiently. Additionally, text summarization enhances information retrieval systems by providing users with condensed summaries that capture the essence of the original content.

CHAPTER - 1

INTRODUCTION

Text Summarization using NLP is a rapidly evolving field that plays a crucial role in condensing large volumes of text into concise and informative summaries. With the exponential growth of digital information, the ability to extract key information efficiently has become increasingly important. Text summarization techniques leverage the power of Natural Language Processing (NLP) to automatically generate summaries that capture the essence of the original text while reducing the time and effort required for information consumption.

This project aims to explore and implement text summarization strategies using NLP techniques, with a focus on extractive and abstractive methods. Extractive methods involve selecting significant sentences or phrases from the original text, while abstractive methods generate new sentences that capture the main ideas. By comparing the performance and effectiveness of these approaches, we aim to identify the most suitable strategy for the given task.

One of the primary challenges in text summarization is maintaining coherence and preserving the essential information while reducing the text length. Coherence refers to the logical flow and connectivity of the summary, ensuring that it reads naturally and makes sense to the reader. Additionally, domain-specific language and knowledge need to be considered to ensure that the generated summaries are accurate and relevant within specific domains such as news, scientific literature, or legal documents.

Advancements in machine learning and deep learning have significantly contributed to the development of high-quality summarization models. These models leverage large-scale datasets and sophisticated algorithms to learn the patterns and relationships within the text, enabling them to generate coherent and informative summaries. By harnessing the power of NLP models such as BERT, GPT, or Transformer-based architectures, we can achieve state-of-the-art performance in text summarization tasks.

The project will utilize popular NLP libraries such as Spacy and Streamlit, along with the NewsAPI as a data source, to implement the text summarization system. Spacy provides a wide range of NLP functionalities, including part-of-speech tagging, named entity recognition, and dependency parsing, which are essential for understanding the structure and content of the text. Streamlit, on the other hand, will be used to develop a user-friendly interface that allows users to interact with the summarization system effortlessly.

The applications of text summarization using NLP are vast and diverse. News aggregation platforms can utilize text summarization to present users with key points from multiple news articles, enabling them to stay informed in a time-efficient manner. Document summarization aids in quickly understanding the main ideas of lengthy documents, facilitating efficient information extraction. Moreover, text summarization can enhance information retrieval systems by providing users with condensed summaries that capture the essence of the original content, helping them make informed decisions and consume information more effectively.

1.1 EXISTING WORKS

Transformer-based models: Transformer-based models, such as BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer), have shown remarkable performance in various NLP tasks, including text summarization. These models leverage self-attention mechanisms to capture contextual relationships between words and generate coherent and informative summaries.

Seq2Seq models: Sequence-to-Sequence (Seq2Seq) models, often based on recurrent neural networks (RNNs) or transformer architectures, have been widely used for text summarization. These models consist of an encoder that reads the input text and a decoder that generates the summary. They can be trained using techniques like teacher forcing or reinforcement learning to optimize the summarization process.

Pointer-Generator networks: Pointer-Generator networks combine extractive and abstractive methods by allowing the model to copy words or phrases from the source text while also generating new words. These models use attention mechanisms to identify important information in the input text and generate summaries that are both coherent and faithful to the original content.

Reinforcement Learning-based models: Reinforcement Learning (RL) has been applied to text summarization to optimize the summarization process. RL-based models use reward signals to guide the generation of summaries, allowing the model to learn to produce high-quality summaries through trial and error.

CHAPTER 2

PROJECT WORK

2.1 OBJECTIVE OF THE PROJECT

2.1.1. Develop a robust and accurate text summarization system

The objective of developing a robust and accurate text summarization system is to create a system that can effectively condense large texts into concise summaries while preserving the essential information. This requires employing NLP techniques for feature extraction, language modeling, and evaluation metrics. By considering extractive or abstractive strategies, preprocessing the text, incorporating domain-specific considerations, optimizing for efficiency and scalability, and designing a user-friendly interface, the system can generate coherent and informative summaries. Such a system has the potential to greatly enhance information consumption, decision-making, and overall efficiency across various domains.

2.1.2. Explore and implement different summarization strategies

In order to enhance the text summarization process, it is crucial to explore and implement different summarization strategies. This objective involves investigating various approaches, such as extractive and abstractive methods, and evaluating their performance and effectiveness. By comparing the outcomes of these strategies, we can identify the most suitable approach for the given task. This exploration and implementation of different summarization strategies contribute to the advancement of text summarization research and enable the development of more accurate and efficient summarization systems.

2.1.3. Evaluate the quality and coherence of generated summaries

Evaluating the quality and coherence of generated summaries is a crucial step in the text summarization process. This objective involves developing evaluation metrics and techniques to measure the accuracy, relevance, and coherence of the summaries produced by the system. By assessing the performance of the system against these metrics, we can gain insights into the strengths and limitations of the summarization model. This evaluation process helps in understanding how well the system captures the essential information and maintains the logical flow of the original text. By continuously refining and improving the evaluation methods, we can ensure that the generated summaries meet the desired standards of quality and coherence.

2.1.4. Incorporate domain-specific language and knowledge

Incorporating domain-specific language and knowledge is a crucial objective in text summarization. This involves leveraging specialized dictionaries, ontologies, or domain-specific word embeddings to enhance the accuracy and relevance of generated summaries within specific domains. By incorporating domain-specific language and knowledge, we can ensure that the summaries capture the nuances and terminology specific to the target domain. This improves the overall quality and effectiveness of the summarization system, making it more valuable and applicable in domains such as news, scientific literature, or legal documents.

2.1.5. Optimize the summarization model for efficiency and scalability

It is one of the key objectives in text summarization. This involves exploring techniques to improve the processing speed and memory usage of the system, enabling it to handle large volumes of text data effectively. By optimizing the model, we can ensure that the summarization process is fast and resource-efficient, making it suitable for real-time or near real-time applications. Techniques such as parallel processing, distributed computing, or model compression can be employed to optimize the summarization process and enhance its scalability. This objective aims to develop a system that can handle large-scale text data efficiently, making it applicable in various scenarios where timely and efficient summarization is required.

2.2 LITERATURE SURVEY

● [1] As reported by Mingyang Geng, Shangwen Wang, Dezun Dong, Haotian Wang, Shaomeng Cao, Kechi Zhang, Zhi Jin, this model uses a Two-stage interpretation process and offers a 30% increase in off-the-shelf models. But it deviates a lot and leaves room for future work which is left unfinished.

● [2] As stated by Aakash Bansal, Chia-Yi Su, Collin McMillan, the model makes use of a neural model for source code summarization that augments a standard Transformer encoder/decoder architecture using subroutines. There's no clear consensus on whether it creates more concise summaries than baseline, and file context methods are a bit unstable.

● [3] Chia-Yi Su and Collin McMillan used a model where they fine-tune, distill, and re-train the GPT model. This improves the GPT model's performance for source code summarization. This research deprecates BLEU and replaces it with USE, which is a metric that encodes the reference and the predicted summary to a fixed-length vector by using a universal encoder and computes the similarity scores between two summaries. Whereas BLEU only considers word overlap. However, this model has high training time.

● [4] Yao Wan, Zhou Zhao, Min Yang, Guandong Xu, Haochao Ying, Jian Wu, Philip S. Yu use reinforcement learning for better decoding with Ground Truth. The LSTM used in this model has very low efficiency.

● [5] Wenhua Wang, Yuqun Zhang, Yulei Sui, Yao Wan, Zhou Zhao, Jian Wu, Philip S. Yu, and Guandong Xu also use a reinforcement learning model, but they use a Hierarchical attention network with a ROUGE-L score of 45.23. No secondary checks are done, and there's a lack of context awareness.

● [6] In the year 2021, Yumo Xu and Mirella Lapata conducted research on document summarization with latent queries. They utilized a latent query model based on query-based representations. The dataset used in this study is the CNN/Daily Mail dataset, and the F1 ROUGE score achieved is 48.5.

● [7] In 2023, Mingyang Geng, Shangwen Wang, Dezun Dong, Haotian Wang, Ge Li, Zhi Jin, Xiaoguang Mao, and Xiangke Liao focused on an empirical study on using large language models for multi-intent comment generation. They applied their model to Funcom, TLC, and OpenAI Codex data. The evaluation metrics used are ROUGE-L, METEOR, and BLEU, with a TLC - ROUGE-L score of 90.8.

● [8] In 2020, Menaka Pushpa Arthur worked on automatic source code documentation using a code summarization technique of NLP. The paper mentions using a software word usage model and random code snippets. The Rouge-1, L, W, and Rouge-L score achieved is 78.27. However, there were issues with the dataset and potential overfitting for test data.

● [9] In 2021, Ziyi Zhou, Huiqun Yu, and Guisheng Fan explored adversarial training and ensemble learning for automatic code summarization. They used GANs and ensemble learning, employing extracted Java code pairs from GitHub. The evaluation metrics include BLEU-4, METEOR, and ROUGE-L, with a Rouge-L score of 49.37. This approach may have limitations related to hyperparameter selection and tuning, potentially making applications limited and narrow.

● [10] In 2022, Pratik K Biswas and Aleksandr Iakubovich conducted extractive summarization of call transcripts. This research focused on call transcripts and was specifically designed for text and conversations, with a ROUGE-L score of 76.18.

2.3 SYSTEM ARCHITECTURE AND MODULES

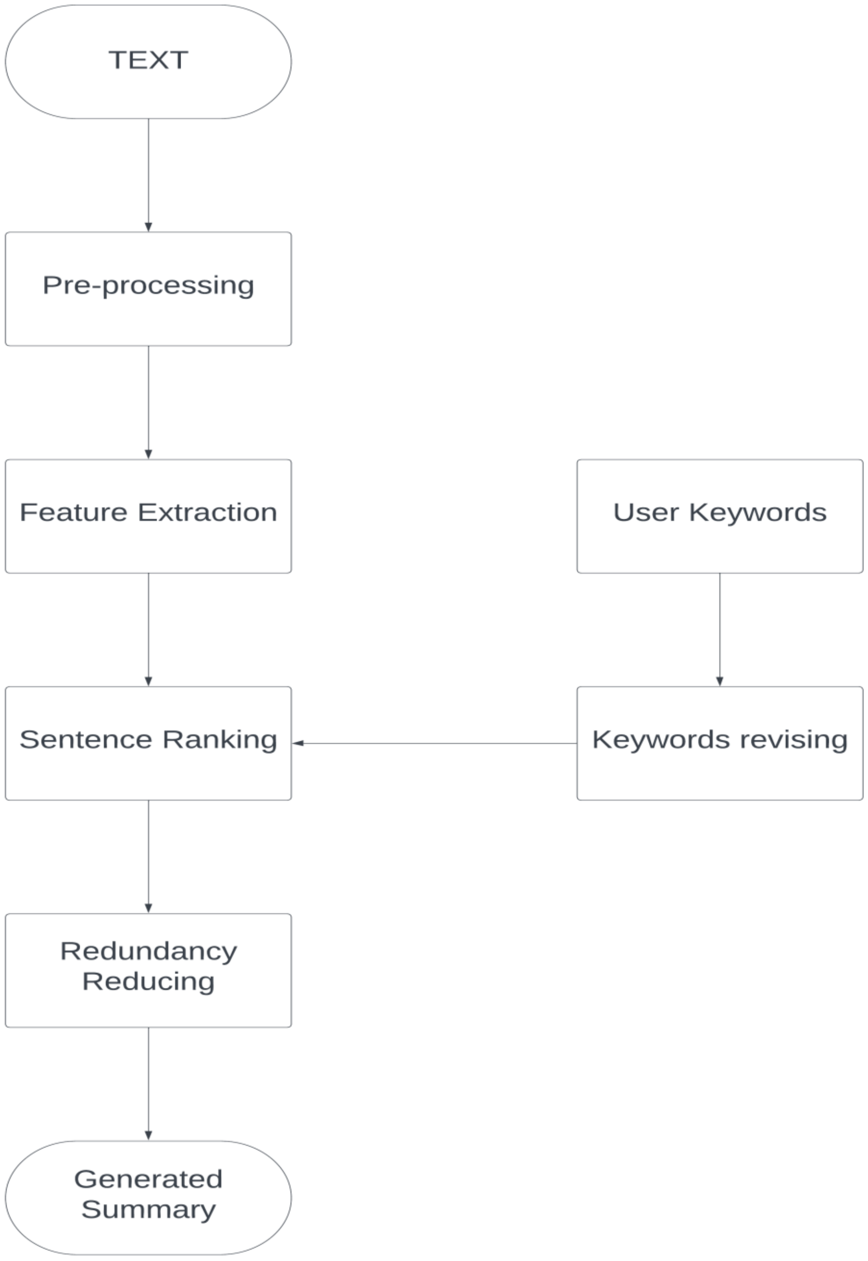


Figure 2.1 - Architecture Diagram

The approach for this project involves a systematic and iterative process to develop an effective text summarization system. The following steps outline the approach:

1. **Data collection:** The first step is to gather a diverse and representative dataset of text documents. This dataset will serve as the basis for training and evaluating the summarization model.

2. **Preprocessing:** The collected text data needs to be pre-processed to remove noise, perform tokenization, and handle any specific requirements of the summarization model. This step ensures that the data is in a suitable format for further analysis.

3. **Model selection:** The next step is to select an appropriate model for text summarization. This can include traditional algorithms like TextRank or more advanced deep learning models such as transformer-based architectures like BERT or GPT. The choice of model depends on the specific requirements and constraints of the project.

4. **Training:** Once the model is selected, it needs to be trained on the pre-processed dataset. This involves feeding the model with input-output pairs, where the input is the original text and the output is the corresponding summary. The model learns to generate summaries by optimizing certain objective functions, such as minimizing the difference between the generated summary and the reference summary.

5. **Evaluation:** After training, the model needs to be evaluated to assess its performance. This can be done by comparing the generated summaries with human-generated reference summaries using evaluation metrics such as ROUGE (Recall-Oriented Understudy for Gisting Evaluation). The evaluation helps in understanding the strengths and weaknesses of the model and guides further improvements.

6. **Iterative refinement:** Based on the evaluation results, the model can be refined and fine-tuned to improve its performance. This may involve adjusting hyperparameters, incorporating additional features or techniques, or collecting more data for training.

7. **Deployment:** Once the model achieves satisfactory performance, it can be deployed to generate summaries for new, unseen text data. This can be done through a user-friendly interface or integrated into existing systems for information retrieval or content analysis.

8. **Continuous evaluation and improvement:** The development of a text summarization system is an ongoing process. It is important to continuously evaluate the system's performance and gather user feedback to identify areas for improvement. This can involve conducting user studies, collecting additional data, or exploring new techniques and algorithms. By iteratively refining the system based on feedback and emerging research, the summarization model can evolve and adapt to changing user needs and advancements in the field.

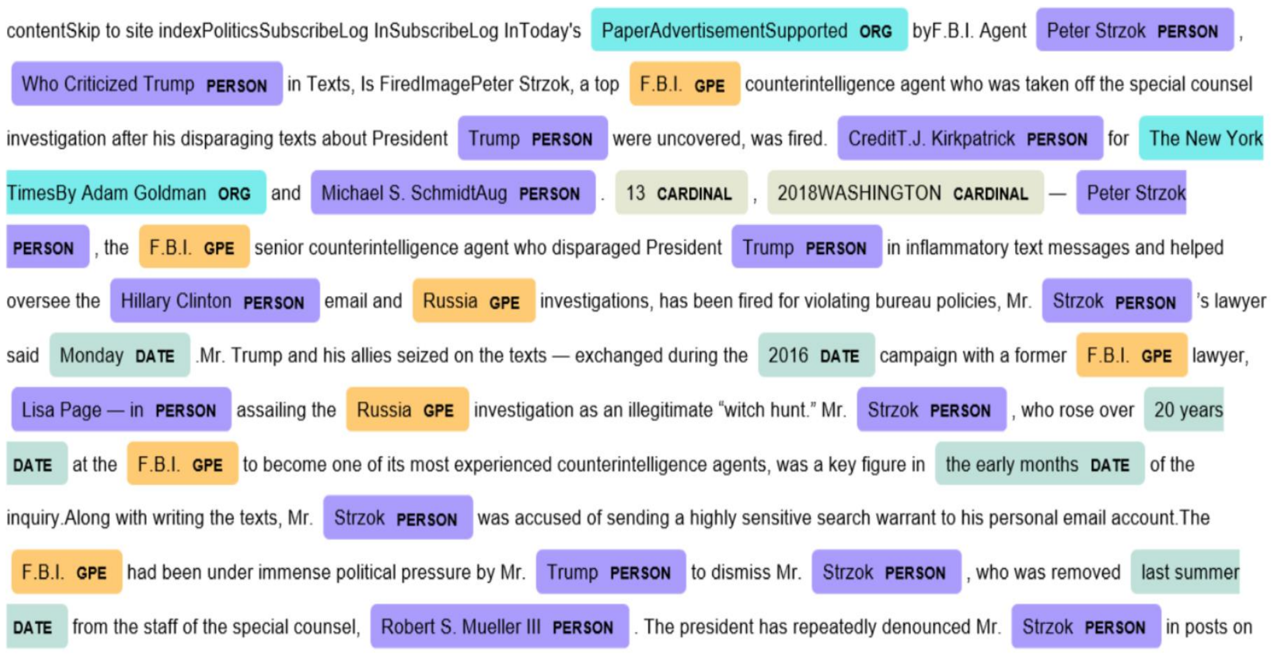
2.3.1 Machine Learning Models

To summarize the given text, various machine learning techniques are applied to preprocess data. Models that are commonly used include:

TextRank Algorithm: TextRank is an unsupervised graph-based ranking algorithm that has been widely used for text summarization tasks. Inspired by the PageRank algorithm used in web search, TextRank treats sentences or words in a document as nodes in a graph and establishes connections between them based on their co-occurrence patterns. By iteratively calculating the importance scores of each node in the graph, TextRank identifies the most significant sentences or words that represent the key information in the document. These important nodes are then used to construct a summary by selecting the top-ranked sentences or words. TextRank's strength lies in its simplicity and effectiveness in capturing the salient information in a document without the need for extensive training data. It has been successfully applied in various domains and has proven to be a valuable tool for automatic text summarization.

Parts-Of-Speech Tagging: Parts-of-speech tagging, also known as POS tagging, is a fundamental task in natural language processing (NLP) that involves assigning grammatical labels to words in a sentence. The goal of POS tagging is to determine the syntactic category of each word, such as noun, verb, adjective, or adverb, among others. This process is essential for understanding the structure and meaning of a sentence, as it provides valuable information about the role and function of each word within the context. POS tagging algorithms utilize various techniques, including rule-based approaches, statistical models, and machine learning algorithms, to accurately assign the appropriate part-of-speech tags to words. By performing POS tagging, NLP systems can extract valuable insights from text, enabling a wide range of applications such as information extraction, sentiment analysis, and machine translation.

Name Entity Recognition: It is a crucial task in natural language processing (NLP) that involves identifying and classifying named entities in text. Named entities refer to real-world objects such as persons, organizations, locations, dates, and more. NER algorithms aim to accurately recognize and classify these entities to provide valuable information about the text's content. By leveraging various techniques such as rule-based approaches, statistical models, and machine learning algorithms, NER systems can effectively identify and categorize named entities within a given text. This information is particularly useful in applications such as information extraction, question answering, and text summarization, where understanding and extracting specific entities play a vital role in generating meaningful insights and facilitating further analysis. NER is a critical component in many NLP pipelines, enabling systems to comprehend and process text at a deeper level.



contentSkip to site indexPoliticsSubscribeLog InSubscribeLog InToday's PaperAdvertisementSupported ORG byF.B.I. Agent Peter Strzok PERSON , Who Criticized Trump PERSON in Texts, Is FiredImagePeter Strzok, a top F.B.I. GPE counterintelligence agent who was taken off the special counsel investigation after his disparaging texts about President Trump PERSON were uncovered, was fired. CreditT.J. Kirkpatrick PERSON for The New York TimesBy Adam Goldman ORG and Michael S. SchmidtAug PERSON . 13 CARDINAL , 2018WASHINGTON CARDINAL — Peter Strzok PERSON , the F.B.I. GPE senior counterintelligence agent who disparaged President Trump PERSON in inflammatory text messages and helped oversee the Hillary Clinton PERSON email and Russia GPE investigations, has been fired for violating bureau policies, Mr. Strzok PERSON 's lawyer said Monday DATE .Mr. Trump and his allies seized on the texts — exchanged during the 2016 DATE campaign with a former F.B.I. GPE lawyer, Lisa Page — in PERSON assailing the Russia GPE investigation as an illegitimate "witch hunt." Mr. Strzok PERSON , who rose over 20 years DATE at the F.B.I. GPE to become one of its most experienced counterintelligence agents, was a key figure in the early months DATE of the inquiry.Along with writing the texts, Mr. Strzok PERSON was accused of sending a highly sensitive search warrant to his personal email account.The F.B.I. GPE had been under immense political pressure by Mr. Trump PERSON to dismiss Mr. Strzok PERSON , who was removed last summer DATE from the staff of the special counsel, Robert S. Mueller III PERSON . The president has repeatedly denounced Mr. Strzok PERSON in posts on

Figure 2.2 – Named Entity Recognition

Model Training:

In order to develop machine learning and deep learning models, model training is an essential first step. It describes how a model is trained to make predictions or decisions by subjecting it to a labeled dataset and allowing the model to learn from the information. In the process of training, the model modifies its internal parameters, also known as weights, in an effort to reduce the difference between the target values in the training data and its predictions. Building a model that can generalize its knowledge to produce precise predictions on fresh, unobserved data is the main objective of model training.

Validation and Evaluation:

Validation: During the validation step, a separate dataset, known as the validation set, is used to evaluate the model's performance. This dataset is distinct from the training data and provides an unbiased assessment of how well the model generalizes to new, unseen data. By measuring various metrics, such as accuracy, precision, recall, or F1 score, we can determine how well the model performs on the validation set. This step helps in identifying any potential issues, such as overfitting or underfitting, and allows us to fine-tune the model accordingly.

Evaluation: Once the model has been validated, the evaluation step is performed to assess its performance on a completely independent dataset, known as the test set. This dataset is kept separate throughout the entire model development process and is used to provide an unbiased evaluation of the model's performance in real-world scenarios. By comparing the model's predictions with the ground truth labels or values in the test set, we can measure its accuracy, error rate, or any other relevant metric. This evaluation helps in determining the model's overall effectiveness and its ability to fulfil its intended function.

The choice of evaluation metrics depends on the specific task and the desired outcome. For classification tasks, metrics such as accuracy, precision, recall, and F1 score are commonly used. In regression tasks, metrics like mean squared error (MSE) or mean absolute error (MAE) are often employed.

2.3.2. CONSIDERED APPROACHES

2.3.2.1. Recurrent Neural Networks

For time series data, RNNs and their variations, such as Long Short-Term Memory (LSTM) networks, work well. They are appropriate for short-term forecasting and can capture sequential dependencies. Particulate matter, ozone, carbon monoxide, and other pollutants, as well as meteorological elements like temperature, humidity, and wind speed, are just a few examples of the numerous air quality indicators that are forecasted at different places across time. When modeling the dynamic nature of air quality data, RNNs work well because they can capture temporal relationships and patterns in the data. We can adjust hyperparameters such as learning rate, batch size, and the number of epochs to optimize the model's performance.

2.3.2.2. Convolutional Neural Networks

CNNs can be used with spatial data, such as images of air quality patterns. They extract spatial features from input data and can be combined with other models for forecasting.. CNNs may be used to handle both temporal and spatial correlations in air quality datasets, however they are less straightforward to use than RNNs for sequential data. Convolutional layers are usually used in CNN architectures for feature extraction, and then pooling layers are added to minimize spatial dimensions and prevent overfitting. The output is smoothed and then sent via fully linked layers for prediction. For better performance, residual connections or more advanced CNN structures could be used.

2.3.2.3. Bidirectional Encoder Representations from Transformer

BERT (Bidirectional Encoder Representations from Transformers) is a state-of-the-art transformer-based model in natural language processing (NLP). It has revolutionized various NLP tasks, including text classification, named entity recognition, question answering, and, importantly, text summarization. BERT is a pre-trained model that learns contextual representations of words by training on a large corpus of text data. It utilizes a transformer architecture, which allows it to capture the relationships between words in a sentence by considering both the left and right context. This bidirectional approach enables BERT to understand the meaning and context of words more accurately. By fine-tuning BERT on specific downstream tasks, such as text summarization, it can generate high-quality summaries by leveraging its contextual understanding of the input text. BERT has significantly advanced the field of NLP and has become a go-to model for many text-related applications, including text summarization, due to its exceptional performance and ability to capture intricate linguistic patterns.

2.3.3. PROGRAMMING LANGUAGES

2.3.3.1. Python

Figure 2.3 - Python

Python is a high-level, adaptable, and interpreted programming language that is well-known for its readability and simplicity. Python, developed by Guido van Rossum and initially released in 1991, has since become one of the most popular programming languages for a wide range of applications, including web development and data analysis, as well as artificial intelligence and scientific computing. Python's flexibility, simplicity, and strong community support make it a popular choice for a wide range of applications, and it remains a beneficial language for both novice and expert developers.

2.3.4. PYTHON LIBRARIES

2.3.4.1. Spacy

spaCy

Fig 2.4 - spaCy

SpaCy is a popular and powerful Python library for natural language processing (NLP) tasks. It provides a wide range of functionalities and tools that make it easier to work with text data. SpaCy offers efficient and fast processing capabilities, making it suitable for both small-scale and large-scale NLP tasks.

One of the key features of SpaCy is its ability to perform various NLP tasks, such as part-of-speech tagging, named entity recognition, dependency parsing, and sentence segmentation. These tasks are essential for understanding the structure and meaning of text, enabling more advanced analysis and information extraction.

SpaCy also provides pre-trained models for different languages, allowing users to perform NLP tasks without the need for extensive training data. These models have been trained on large corpora and are continuously updated and improved by the SpaCy community.

Another advantage of SpaCy is its user-friendly interface and intuitive API, which makes it easy to integrate into Python applications. It provides a streamlined and efficient workflow for processing text data, allowing users to focus on the analysis and interpretation of the results.

Additionally, SpaCy offers support for custom rule-based matching, allowing users to define their own patterns and rules for text extraction and manipulation. This flexibility makes it a versatile tool for a wide range of NLP applications.

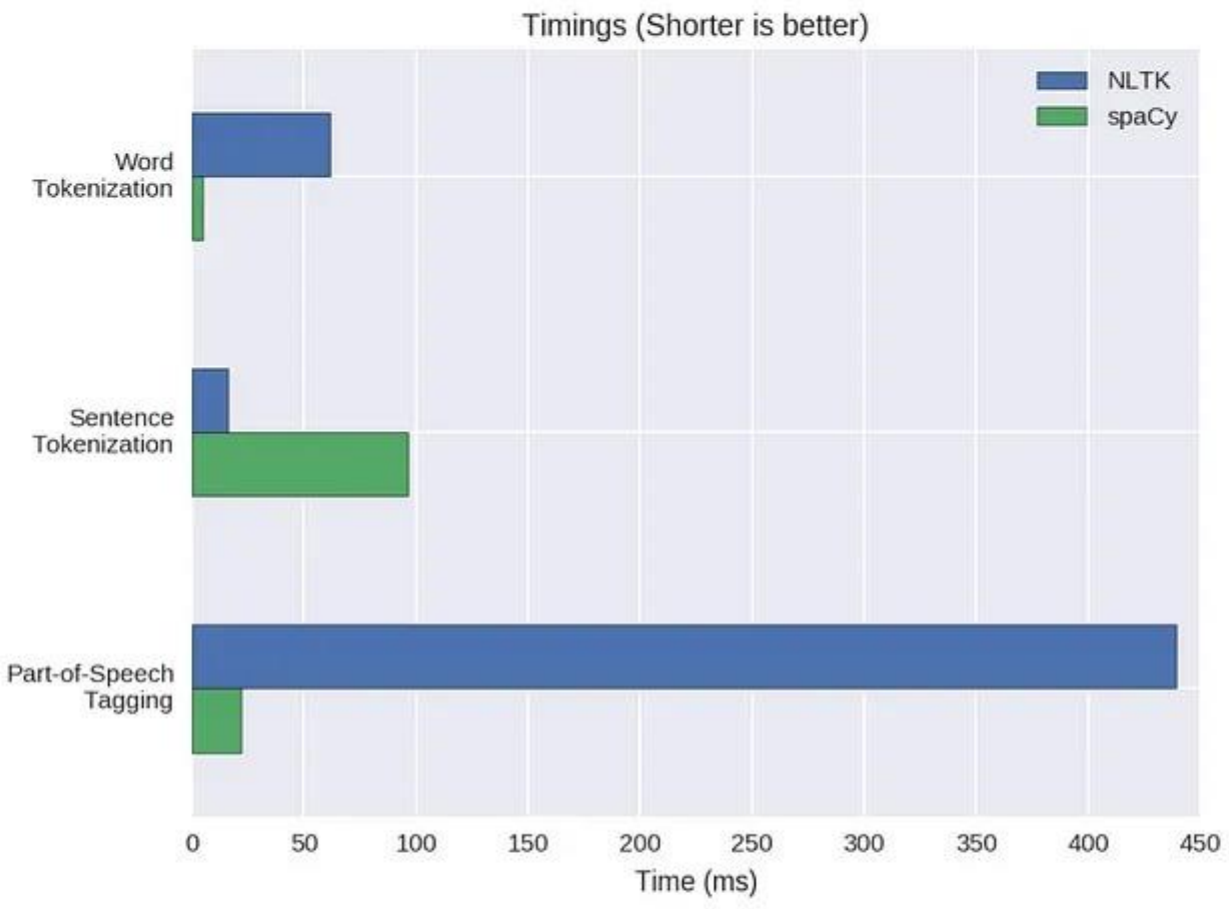


Fig 2.4 – spaCy vs NLTK

Abbreviation:

NLTK: Natural Language ToolKit

2.3.4.2. Streamlit

Fig 2.5 - Streamlit

Streamlit is a Python library that simplifies the process of building interactive web applications for data science and machine learning projects. It allows developers to create and deploy data-driven applications quickly and easily, without the need for extensive web development knowledge.

With Streamlit, developers can write code in a simple and intuitive manner, using familiar Python syntax. It provides a clean and declarative API that enables the creation of interactive visualizations, data exploration tools, and dashboards with just a few lines of code.

Streamlit also offers a wide range of built-in components and widgets, such as sliders, dropdowns, and checkboxes, that make it easy to add interactivity to the applications. These components can be used to control the behaviour and parameters of the underlying data analysis or machine learning models, allowing users to dynamically explore and interact with the results.

One of the key features of Streamlit is its automatic hot-reloading capability. This means that any changes made to the code are immediately reflected in the running application, making the development and testing process fast and efficient.
Furthermore, Streamlit provides seamless integration with popular Python libraries such as Pandas,

Matplotlib, and Plotly, allowing developers to leverage the power of these libraries for data manipulation and visualization within their Streamlit applications.

2.3.4.3. Newsapi

News API

Fig 2.6 - Newsapi

NewsAPI is a Python client library that provides easy access to a vast collection of news articles and headlines from various sources worldwide. It simplifies the process of fetching and retrieving news data, allowing developers to integrate news content into their applications seamlessly.

With the NewsAPI client library, developers can make API requests to retrieve news articles based on specific criteria such as keywords, sources, language, or date range. This flexibility enables users to tailor their news content to their specific needs and preferences.

The library provides a straightforward and intuitive interface for interacting with the NewsAPI service. Developers can easily authenticate their requests using an API key, ensuring secure and authorized access to the news data.

NewsAPI offers a wide range of functionalities, including the ability to search for news articles, retrieve top headlines, and filter news based on various parameters. This allows developers to create dynamic and personalized news experiences for their users.

2.3.4.4. Gensim

Fig 2.7 - Gensim

Gensim is a popular Python library for topic modeling and natural language processing (NLP) tasks. It provides a simple and efficient way to analyze and extract insights from large text corpora.

One of the key features of Gensim is its implementation of various topic modeling algorithms, such as Latent Dirichlet Allocation (LDA) and Latent Semantic Analysis (LSA). These algorithms allow users to discover latent topics within a collection of documents and understand the underlying themes and patterns in the data. It also offers functionality for document similarity analysis, allowing users to compare and find similarities between documents based on their content. This can be useful for tasks such as document clustering, recommendation systems, and information retrieval.

Another important aspect of Gensim is its support for word vector representations, such as Word2Vec and FastText. These models enable users to represent words as dense vectors in a high-dimensional space, capturing semantic relationships and similarities between words. This can be leveraged for tasks like word embeddings, semantic search, and text classification.

Gensim provides an intuitive and easy-to-use API, making it accessible for both beginners and experienced NLP practitioners. It offers efficient implementations that can handle large-scale text data, allowing for scalable and high-performance analysis.

2.3.4.5. Sumy

Sumy is a Python library that provides a simple and efficient way to perform automatic text summarization. It offers various algorithms and methods for extracting key information from text documents and generating concise summaries.

One of the key features of Sumy is its support for multiple summarization algorithms, including LexRank, LSA (Latent Semantic Analysis), Luhn, and TextRank. These algorithms employ different techniques, such as graph-based ranking and statistical analysis, to identify important sentences and generate summaries that capture the essence of the original text.

Sumy also provides flexibility in terms of input sources, allowing users to summarize text from various formats, including plain text files, HTML documents, or even web pages by providing the URL. This makes it convenient for processing different types of content and integrating summarization capabilities into various applications.

The library offers a user-friendly API, making it easy to integrate Sumy into existing Python projects. Developers can customize the summarization process by specifying parameters such as the number of sentences or the language of the input text. This allows for fine-tuning the summarization output to meet specific requirements.

Furthermore, Sumy supports multiple languages, enabling users to summarize text in different languages. It utilizes language-specific models and resources to ensure accurate and coherent summaries across various linguistic contexts.

Sumy also provides evaluation metrics to assess the quality of the generated summaries. Developers can compare the output of different algorithms or fine-tune the summarization process based on specific evaluation criteria, such as ROUGE (Recall-Oriented Understudy for Gisting Evaluation) scores.

2.5 METHODOLOGY

The methodology (or) plan for this project involves a multi-step process to create a web application that enables users to summarize news articles from various sources and getting input from the user to summarize the content. The following is the detailed explanation of our work:

1. Streamlit Web Application: The project utilizes Streamlit, a Python library, to develop a user-friendly web application. Streamlit simplifies the process of building interactive web interfaces for data science and machine learning projects. With Streamlit, developers can write code in a simple and intuitive manner using familiar Python syntax. It provides a clean and declarative API that enables the creation of interactive visualizations, data exploration tools, and dashboards with just a few lines of code. By leveraging Streamlit, the project aims to provide users with a seamless and intuitive experience for summarizing news articles.

2. NewsAPI Integration: To fetch news articles, the project integrates with the NewsAPI. NewsAPI is a popular news aggregator that provides access to a vast collection of news articles and headlines from various sources worldwide. By leveraging the NewsAPI, users can retrieve news articles based on their input keywords, categories, or specific sources. This integration allows the web application to provide a diverse range of news articles for summarization.

3. Spacy Text Preprocessing: Before generating summaries, the text of the news articles needs to be pre-processed. The project utilizes the SpaCy, a powerful Python library for natural language processing. SpaCy provides a wide range of tools and functionalities for text preprocessing, including removing stopwords, punctuation, and numbers. By applying these preprocessing techniques, the project aims to clean the text and remove irrelevant information, ensuring that the generated summaries are concise and meaningful.

4. Summarization Algorithms: To generate summaries, our project employs two popular Python libraries: Gensim and Sumy. Gensim is a powerful library for topic modeling and natural language processing, while Sumy focuses specifically on text summarization. These libraries offer various algorithms and methods for extracting key information from text documents and

generating concise summaries. The project utilizes algorithms such as Latent Semantic Analysis (LSA), Latent Dirichlet Allocation (LDA), LexRank, and others. By leveraging these algorithms, the project aims to identify important sentences and capture the essence of the original news articles in the generated summaries.

5. Custom Text Summarization: In addition to summarizing news articles, the project allows users to input their own custom text and choose a summarization method. This feature provides flexibility and customization options for users who want to summarize their own text documents or articles. By incorporating this functionality, the web application caters to a wider range of users and use cases. This step also shows the parts-of-speech tagging (POS)

CHAPTER - 3

3. SOURCE CODE AND RESULTS

3.1: Main Code (app.py)

#Import libraries

```
import streamlit as st
from helper import get_summary, spacy_rander, fetch_news, fetch_news_links
```

# Set up the configuration for the Streamlit web app

```
st.set_page_config(
    page_title=”Text Summarization Web App”,
    page_icon=”🔥”,
    layout=”wide”,
    initial_sidebar_state=”expanded”,
    menu_items={
         ‘About’: ‘This is a header. This is an extremely cool app!’
    }
)
```

# Display the sidebar title and options

```
st.sidebar.title("Text Summarization Web App")
option = ["News Summary and Headlines", "Custom Text Summarization"]
choice = st.sidebar.selectbox("Select your choice", options=option)
```

# Custom Text Summarization option

```
if choice == "Custom Text Summarization":
    st.sidebar.markdown("Copy and paste your text in the text area below to get a summary.")
    st.title("Welcome to Custom Text Summarization")

    col1, col2 = st.columns(2)

    with col1:
        text = st.text_area(label="Enter your text", height=350, placeholder="Enter your text or article 
here")

    if st.button("Get Summary"):
        summary = get_summary(text)
        try:
            with col2:
                st.write("Text Summary:")
                st.code(summary)
                st.write("Text Headline:")
                st.code("Feature Coming Soon")

            spacy_rander(summary)

            # Get the original article analysis
            # with st.expander("Get Original Article Analysis"):
            spacy_rander(text, text="Yes")

        except NameError:
            pass
```

# News Summary and Headlines option

```
if choice == "News Summary and Headlines":
    st.title("BBC News Summary")

    search_query = st.text_input("", placeholder="Enter the topic you want to search")
    st.write(" ")

    link, title, thumbnail = fetch_news_links(search_query)
    fetch_news = fetch_news(link)

    if link != []:
        col1, col2 = st.columns(2)

        with col1:
            for i in range(len(link)):
                if (i % 2) == 0:
                    st.image(thumbnail[i])
                    st.write(title[i])
                    with st.expander("Read The Summary"):
                        st.write(get_summary(fetch_news[i]))
                    st.markdown("[**Read Full Article**]({})".format(link[i]), unsafe_allow_
html=True)
                    st.write(" ")

        with col2:
            for i in range(len(link)):
                if (i % 2) != 0:
                    st.image(thumbnail[i])
                    st.write(title[i])
                    with st.expander("Read The Summary"):
                        st.write(get_summary(fetch_news[i]))
                st.markdown("[**Read Full Article**]({})".format(link[i]), unsafe_allow_
html=True)
                st.write(" ")

   else:
       st.info("No results found for {}. Please try some popular keywords.".format(search
_query)
```

3.2: Support code (helper.py)

```
#Import libraries
import requests
from bs4 import BeautifulSoup
import spacy
from heapq import nlargest
import random
import streamlit as st

# Function to calculate word frequencies in a document
def word_frequency(doc):
    word_frequencies = {}

    for word in doc:

        # Check if the word is not a stopword
        if word.text.lower() not in stopwords:

            # Check if the word is not a punctuation
            if word.text.lower() not in punctuation:

            # If the word is not already in the dictionary, add it with a frequency of 1
                if word.text not in word_frequencies.keys():
                    word_frequencies[word.text] = 1

                # If the word is already in the dictionary, increment its frequency by 1
                else:
                    word_frequencies[word.text] += 1

    return word_frequencies
```

```
# Function to calculate sentence scores based on word frequencies
def sentence_score(sentence_tokens, word_frequencies):
    sentence_score = {}

    for sent in sentence_tokens:
        for word in sent:
            # Check if the word is in the word frequencies dictionary
            if word.text.lower() in word_frequencies.keys():
                # If the sentence is not already in the dictionary, add it with the word's
frequency as the score
                if sent not in sentence_score.keys():
                    sentence_score[sent] = word_frequencies[word.text.lower()]
                # If the sentence is already in the dictionary, increment its score by the
word's frequency
                else:
                    sentence_score[sent] += word_frequencies[word.text.lower()]

    return sentence_score

# Function to fetch news links based on a query
@st.cache(allow_output_mutation=False)
def fetch_news_links(query):
    link_list = []
    title_list = []
    thumbnail_list = []

    if query == "":
        reqUrl = "https://newsapi.org/v2/everything?sources=bbc-
news&q=india&language=en&apiKey=ac5568e7ad914659b1d66c0ee6929560".for
mat(news_api_key)
    else:
        reqUrl = "https://newsapi.org/v2/everything?sources=bbc-
news&q={}&language=en&apiKey=ac5568e7ad914659b1d66c0ee6929560".format
(query, news_api_key)

    headersList = {
        "Accept": "*/*",
        "User-Agent": "Thunder Client (https://www.thunderclient.com)"
    }

    payload = ""
    response = requests.request("GET", reqUrl, data=payload,
headers=headersList).text
    response = json.loads(response)

    tw = 0
    for i in range(len(response["articles"])):
        if tw == 10:
            pass
        else:
            if "/news/" in response["articles"][i]["url"] and "stories" not in
response["articles"][i]["url"]:
                link_list.append(response["articles"][i]["url"])
                title_list.append(response["articles"][i]["title"])
                thumbnail_list.append(response["articles"][i]["urlToImage"])
            else:
                pass
            tw += 1

    return link_list, title_list, thumbnail_list
```

# Function to fetch news content based on a list of links

```
@st.cache(allow_output_mutation=False)
def fetch_news(link_list):
    news = []
    news_list = []

    for i in range(len(link_list)):
        news_reqUrl = link_list[i]
        headersList = {
            "Accept": "*/*",
            "User-Agent": "Thunder Client (https://www.thunderclient.com)"
        }

        payload = ""

        news_response = requests.request("GET", news_reqUrl, data=payload,
headers=headersList)
        soup = BeautifulSoup(news_response.content, features="html.parser")
        for para in soup.findAll("div", {"data-component":"text-block"}):
            news.append(para.find("p").getText())
            joinnews = " ".join(news)
            news_list.append(joinnews)
            news.clear()

        return news_list
```

# Function to generate a summary of a given text

```
    def get_summary(text):
        doc = nlp(text)

        word_frequencies = word_frequency(doc)
```

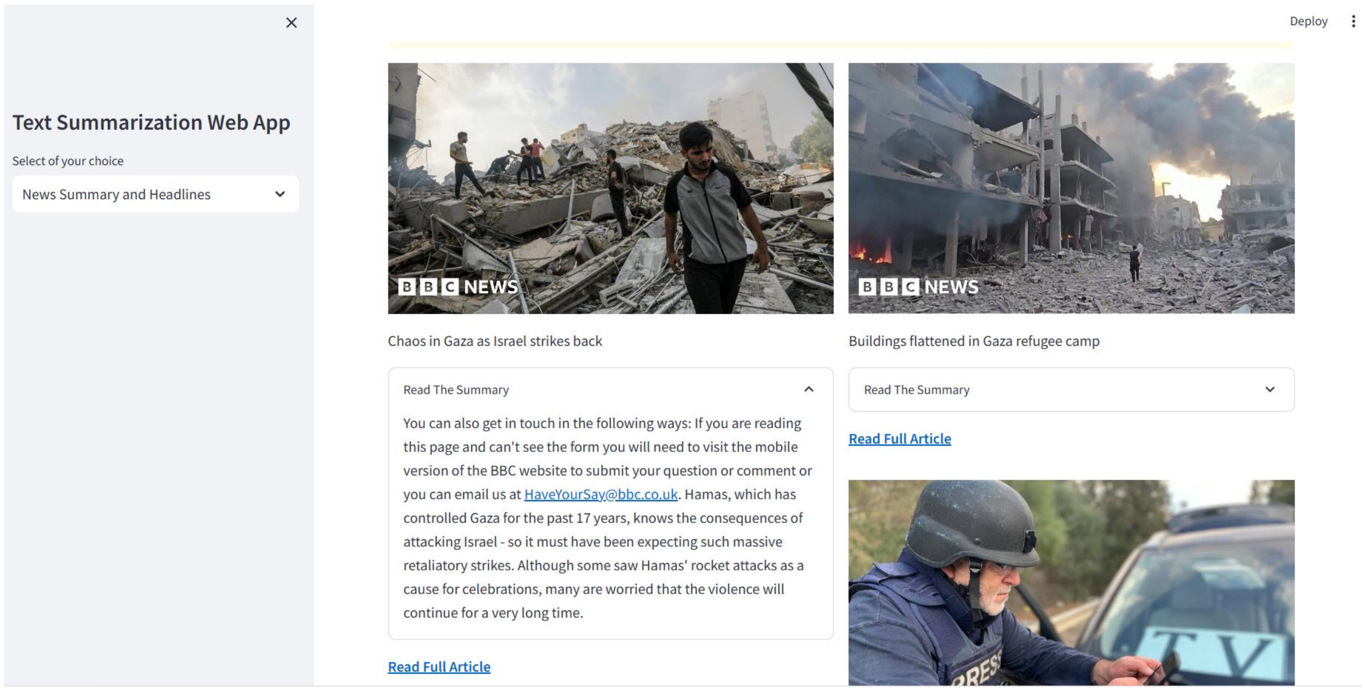
# Normalize the word frequencies by dividing each frequency by the maximum frequency

```
        for word in word_frequencies.keys():
            word_frequencies[word] = word_frequencies[word] /
    max(word_frequencies.values())

        sentence_tokens = [sent for sent in doc.sents]
        sentence_scores = sentence_score(sentence_tokens, word_frequencies)
```

# Select a percentage of the sentences with the highest scores to form the summary

```
        select_length = int(len(sentence_tokens) * 0.10)
        summary = nlargest(select_length, sentence_scores, key=sentence_scores.get)
        summary = [word.text for word in summary]
        summary = " ".join(summary)

        return summary
```

RESULT


Deploy

Text Summarization Web App

Select of your choice

News Summary and Headlines

Chaos in Gaza as Israel strikes back

Read The Summary

You can also get in touch in the following ways: If you are reading this page and can't see the form you will need to visit the mobile version of the BBC website to submit your question or comment or you can email us at HaveYourSay@bbc.co.uk. Hamas, which has controlled Gaza for the past 17 years, knows the consequences of attacking Israel - so it must have been expecting such massive retaliatory strikes. Although some saw Hamas' rocket attacks as a cause for celebrations, many are worried that the violence will continue for a very long time.

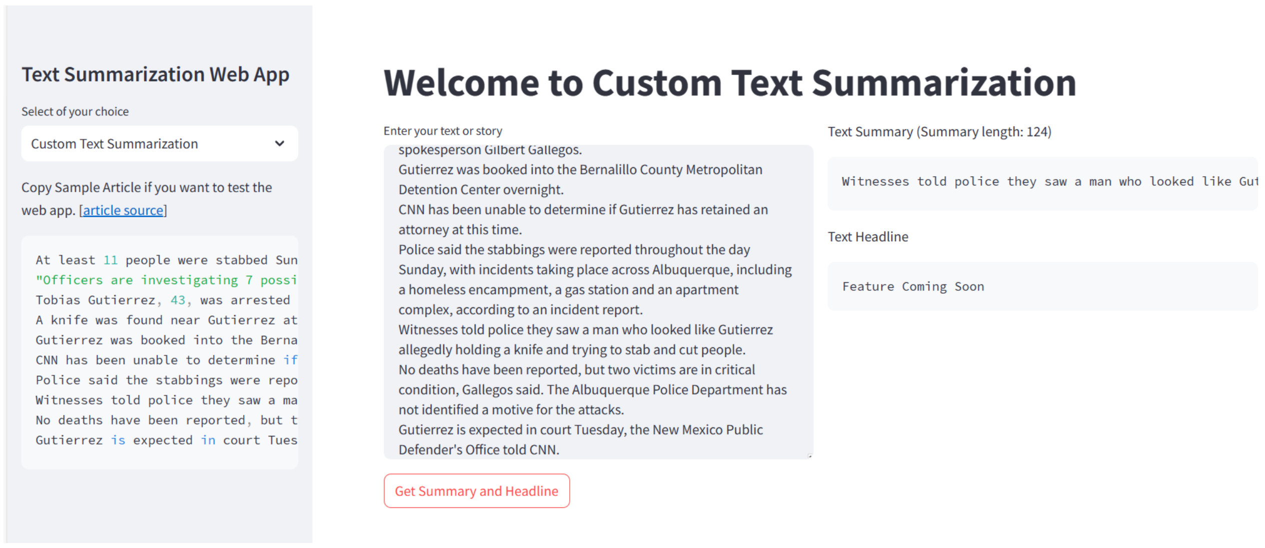
Read Full Article

Buildings flattened in Gaza refugee camp

Read The Summary

Read Full Article

Figure 3.1: News Headlines and Summary


Text Summarization Web App

Select of your choice

Custom Text Summarization

Copy Sample Article if you want to test the web app. [article source]

```
At least 11 people were stabbed Sun
"Officers are investigating 7 possi
Tobias Gutierrez, 43, was arrested
A knife was found near Gutierrez at
Gutierrez was booked into the Berna
CNN has been unable to determine if
Police said the stabbings were repo
Witnesses told police they saw a ma
No deaths have been reported, but t
Gutierrez is expected in court Tues
```

Welcome to Custom Text Summarization

Enter your text or story

spokesperson Gilbert Gallegos.
Gutierrez was booked into the Bernalillo County Metropolitan Detention Center overnight.
CNN has been unable to determine if Gutierrez has retained an attorney at this time.
Police said the stabbings were reported throughout the day Sunday, with incidents taking place across Albuquerque, including a homeless encampment, a gas station and an apartment complex, according to an incident report.
Witnesses told police they saw a man who looked like Gutierrez allegedly holding a knife and trying to stab and cut people.
No deaths have been reported, but two victims are in critical condition, Gallegos said. The Albuquerque Police Department has not identified a motive for the attacks.
Gutierrez is expected in court Tuesday, the New Mexico Public Defender's Office told CNN.

Get Summary and Headline

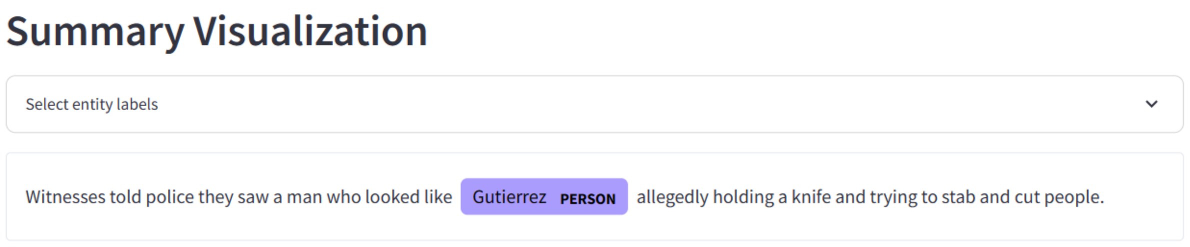
Text Summary (Summary length: 124)

```
Witnesses told police they saw a man who looked like Gut
```

Text Headline

```
Feature Coming Soon
```

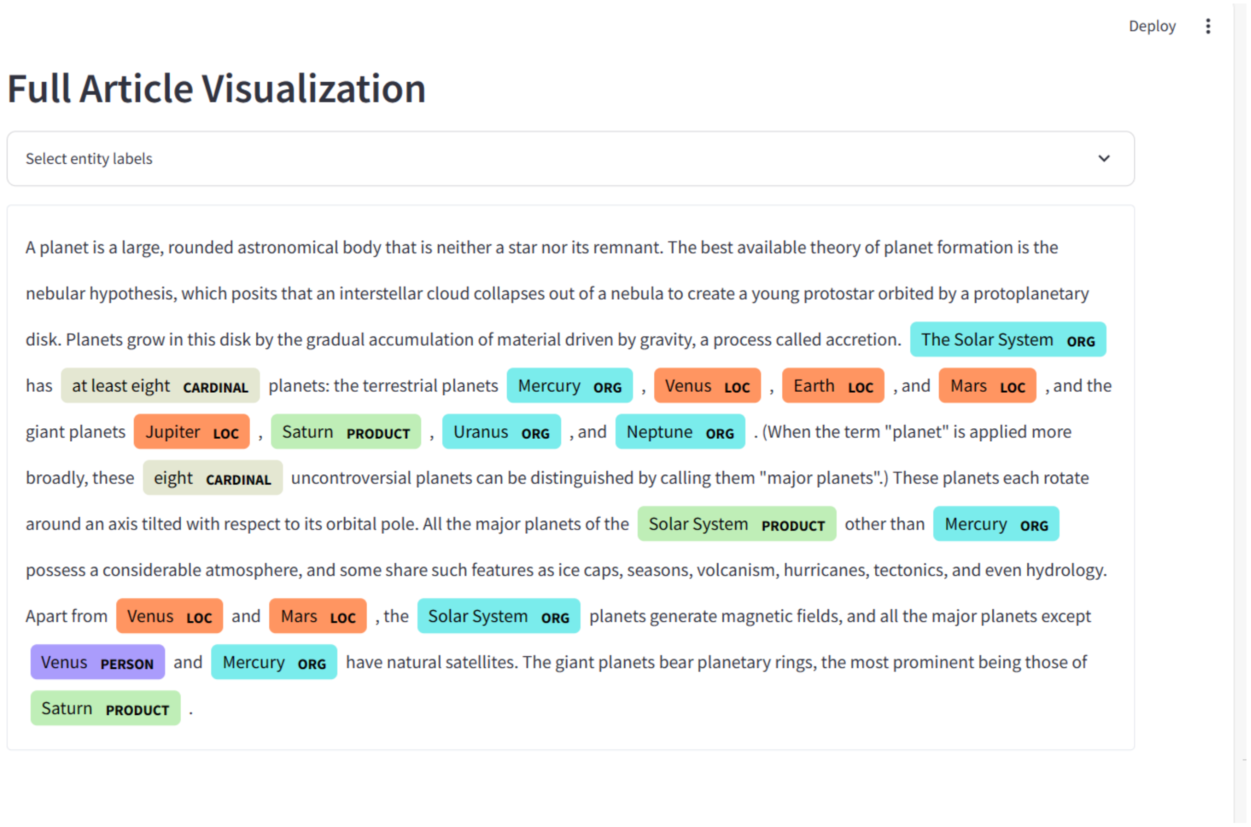
Figure 3.2: Custom Text Summarization


Summary Visualization

Select entity labels

Witnesses told police they saw a man who looked like Gutierrez PERSON allegedly holding a knife and trying to stab and cut people.

Figure 3.3: Summary Visualization


Deploy

Full Article Visualization

Select entity labels

A planet is a large, rounded astronomical body that is neither a star nor its remnant. The best available theory of planet formation is the nebular hypothesis, which posits that an interstellar cloud collapses out of a nebula to create a young protostar orbited by a protoplanetary disk. Planets grow in this disk by the gradual accumulation of material driven by gravity, a process called accretion. The Solar System ORG has at least eight CARDINAL planets: the terrestrial planets Mercury ORG , Venus LOC , Earth LOC , and Mars LOC , and the giant planets Jupiter LOC , Saturn PRODUCT , Uranus ORG , and Neptune ORG . (When the term "planet" is applied more broadly, these eight CARDINAL uncontroversial planets can be distinguished by calling them "major planets".) These planets each rotate around an axis tilted with respect to its orbital pole. All the major planets of the Solar System PRODUCT other than Mercury ORG possess a considerable atmosphere, and some share such features as ice caps, seasons, volcanism, hurricanes, tectonics, and even hydrology. Apart from Venus LOC and Mars LOC , the Solar System ORG planets generate magnetic fields, and all the major planets except Venus PERSON and Mercury ORG have natural satellites. The giant planets bear planetary rings, the most prominent being those of Saturn PRODUCT .

Figure 3.4: Full Article Visualization

CHAPTER - 4

4. CONCLUSION AND FUTURE SCOPE

4.1 CONCLUSION

In conclusion, we have explored various aspects of text summarization using NLP techniques. We discussed different strategies for summarization, such as the TextRank algorithm, parts-of-speech tagging, and named entity recognition. We also examined the importance of model training and evaluation in developing effective summarization models.

Furthermore, we explored the capabilities of popular Python libraries such as spaCy, BERT, Gensim, and Streamlit in implementing text summarization functionality. These libraries provide powerful tools for tasks like word frequency analysis, document similarity, and building interactive web applications for text summarization.

Additionally, we discussed the Thunder Client Python library, which simplifies the process of testing APIs with its lightweight and user-friendly interface. We also touched upon the features of the Streamlit Python library, which enables the creation of interactive web interfaces for data science and machine learning projects.

Overall, text summarization using NLP offers great potential in extracting key information from large volumes of text. It can be applied in various domains such as news articles, research papers, and social media analysis. With the advancements in NLP techniques and the availability of powerful libraries, text summarization has become more accessible and efficient.

Machine learning in air quality prediction and forecasting is a valuable and emerging topic with the potential to significantly improve public health and the environment. In this paper, we review and compare recent studies on the evaluation of air quality using data analytics, machine learning models,and methodologies. We also reflect on our most recent literature analysis. The field of machine learning-based air quality forecasting and prediction, however, still has problems. These challenges include the need for model improvement, the integration of more diverse data sources, and the creation of applications that are user-friendly.

4.2. FUTURE SCOPE

1. Enhanced summarization models: Continued research can lead to the development of more advanced and accurate summarization models, incorporating deep learning techniques and leveraging large-scale pre-trained language models.

2. Multi-modal summarization: Future work can focus on summarizing information from multiple modalities, such as text, images, and videos, to provide more comprehensive summaries.

3. Domain-specific summarization: Customizing summarization models for specific domains can enhance their effectiveness. Training models on domain-specific datasets and incorporating domain-specific knowledge can lead to more relevant and accurate summaries.

4. Evaluation metrics: Developing robust evaluation metrics that capture the quality, coherence, and informativeness of summaries more effectively is an ongoing challenge.

5. Real-time summarization: Future efforts can be directed towards developing efficient and scalable summarization models that can generate summaries in real-time, enabling applications such as news aggregation and social media monitoring.

6. Ethical considerations: Addressing ethical considerations such as bias, fairness, and privacy in text summarization is important. Future research can focus on developing techniques to mitigate biases in summaries and ensure responsible use of summarization models.

7. Multilingual summarization: Expanding text summarization models to handle multiple languages will enable broader access to summarized content across diverse linguistic communities.

In conclusion, the future of text summarization lies in advancing summarization models, exploring multi-modal and domain-specific summarization, improving evaluation metrics, enabling real-time summarization, addressing ethical considerations, and expanding to multilingual summarization. These advancements will contribute to more accurate, efficient, and versatile text summarization system.

5. REFERENCES

[1] Mingyang Geng, Shangwen Wang, Dezun Dong, Haotian Wang, Shaomeng Cao, Kechi Zhang, Zhi Jin. "Interpretation-based Code Summarization" (2023).

[2] Aakash Bansal, Chia-Yi Su, Collin McMillan. "Revisiting File Context for Source Code Summarization" (2023).

[3] Chia-Yi Su, Collin McMillan. "Distilled GPT for Source Code Summarization" (2021). IEEE Access 9.

[4] Yao Wan, Zhou Zhao, Min Yang, Guandong Xu, Haochao Ying, Jian Wu, Philip S. Yu. "Improving Automatic Source Code Summarization via Deep Reinforcement Learning" (2020).

[5] Wenhua Wang, Yuqun Zhang, Yulei Sui, Yao Wan, Zhou Zhao, Jian Wu, Philip S. Yu and Guandong Xu. "Reinforcement-Learning -Guided Source Code Summarization Using Hierarchical Attention" (2020).

[6] Yumo Xu and Mirella Lapata. "Document Summarization with Latent Queries" (2021).

[7] Mingyang Geng, Shangwen Wang, Dezun Dong, Haotian Wang, GeLI, Zhi Jin, Xiaoguang Mao and Xiangke Liao. "An Empirical Study on Using Large Language Models for Multi-Intent Comment Generation" (2023).

[8] Partik K Biswas, Aleksandr lakubovich. "Extractive Summarization of Call Transcripts" (2022)