ARTIFICIAL INTELLIGENCE PROJECT

ON

TEXT SUMMARIZATION USING NLP

SUBMITTED BY

KISHAN SAI SAGUTURU (00951675)
VENKAT REDDY VUNDYALA (00869184)

SUBMITTED TO

Dr. Vahid Behzadan

TABLE OF CONTENTS:

INTRODUCTION:

Text summarization is a vital NLP field focusing on condensing large text volumes into concise summaries. With the rise of digital information, these techniques efficiently extract key insights, saving time and effort. This project explores extractive and abstractive summarization methods—extractive techniques select important text portions, while abstractive techniques generate new sentences encapsulating main ideas. By comparing these approaches, the project aims to determine the most effective strategy for summarization tasks.

PROJECT OBJECTIVE:

This project also focuses on key goals to enhance summarization effectiveness:

Develop a Robust System: Employ NLP techniques for feature extraction and preprocessing to produce coherent summaries.

Explore Summarization Strategies: Compare extractive and abstractive methods to identify the most suitable approach.

Evaluate Quality and Coherence: Use metrics like accuracy and ROUGE scores to ensure logical and relevant summaries.

Incorporate Domain Knowledge: Leverage specialized dictionaries or word embeddings for improved relevance in specific fields.

Optimize for Efficiency: Implement techniques like parallel processing to handle large-scale text data efficiently.

APPROACH:

Approach

1. **Technologies Used:**
    - Python packages: spaCy, BeautifulSoup, and Requests.
    - Streamlit for web development.
2. **Functionalities:**
    - Fetch news articles from BBC using NewsAPI.
    - Summarize user-provided custom text.

3. **Deliverables:**
    - Modular Python scripts (main.py and helper.py) for text processing and summarization.
    - Dual-interface web application for custom text and news summarization.

Goals:

Build a Summarization Framework:
Design a robust framework that efficiently processes text and generates coherent summaries.

Create an Application:
Develop a Streamlit-based web application capable of summarizing both custom text and news articles.

Integrate NLP Techniques:
Use NLP libraries such as spaCy for feature extraction and summarization.

Fetch and Process Data:
Fetch news articles from BBC using NewsAPI, and preprocess them using BeautifulSoup for summarization.

Implement Evaluation Metrics:

Employ metrics like ROUGE scores to assess summary quality and relevance.

Include Tokenization Accuracy: Focus on precision in segmenting text into tokens to improve overall processing accuracy, using precision, recall, and F1 score.

Optimize Performance:
Ensure the application is efficient and scalable for handling large datasets.

Assumptions:

Input Text Structure:

All input text is in English and follows proper grammar and sentence structure. The framework assumes the input length does not exceed 10,000 words for efficient processing.

Tokenization Accuracy:

Tokenization is performed at the word level, ensuring precision in segmenting text into tokens. Accuracy is evaluated using precision, recall, and F1 scores compared to a reference tokenization.

Evaluation Metrics:

Summarization quality is assessed using ROUGE scores with an assumption that reference summaries are available. The metrics evaluate coherence, relevance, and fidelity of the generated summaries.

CODE EXPLANATION

Text Summarizer Framework
This project is a Natural Language Processing (NLP) application designed to summarize custom text and news articles. Below are the main components and their functionalities:

Environment:

- The summarization task involves text input (environment), where the agent (summarizer) generates outputs (summaries). The primary goal is to create concise, coherent summaries that retain the most critical information.

Helper Functions (helper.py)

1.Word Frequency Function

Purpose:

- Calculates word frequencies, excluding stop words and punctuation.

Hyperparameters:

- Stop words: Words ignored during processing.
- Punctuation: Characters excluded from frequency calculations.

Key Steps:

- Iterate through the text and count word occurrences.
- Normalize word frequencies by dividing each frequency by the highest frequency in the document.

2. Sentence Scoring Function

Purpose:

- Assigns scores to sentences based on word importance.

Key Steps:

- Evaluate each word in a sentence using pre-computed word frequencies.
- Aggregate the scores for all words in a sentence.

3.Text Summarization

Purpose:

- Generates a summary by selecting the most important sentences.

Key Steps:

- Divide the text into sentences.
- Score each sentence based on word frequencies.
- Select the top 10% of sentences with the highest scores.

4. News Data Fetching

Functionality:

- fetch_news_links(query): Retrieves article links and metadata using the NewsAPI.
- fetch_news(link_list): Extracts article content from provided links using BeautifulSoup.

Output:

- Links, titles, and content for summarization.

5. Tokenization

Purpose:

- Splits text into individual tokens (words and punctuation).

Key Methods:

- spacy_tokenizer(text): Uses spaCy for advanced tokenization.
- custom_tokenizer(text): Uses regular expressions for basic tokenization.

6. Tokenization Quality Evaluation

Purpose:

- Compares tokenization results between spaCy and a custom tokenizer.

Key Steps:

- Compute metrics: Precision, Recall, F1 Score.

- Use these metrics to evaluate tokenization accuracy.

7. ROUGE Scores

Purpose:

- Measures the similarity between generated summaries and reference summaries.

Metrics:

- ROUGE-1 (unigrams), ROUGE-2 (bigrams), ROUGE-L (longest common subsequence).

Web Application (main.py):

Custom Text Summarization

- Users input text directly into the web interface.
- Summaries are generated using the helper functions.

News Article Summarization

- Users search for a topic.
- The app retrieves relevant articles, generates summaries, and displays links for full articles.

Tokenization Evaluation

- Users enter text for tokenization assessment.
- The app computes and displays metrics (Precision, Recall, F1 Score).

Key Design Choices

Hyperparameters

Sentence Selection:

- Top 10% of sentences are chosen for summarization.

Word Frequency Normalization:

- Ensures frequency values are proportional across sentences.

Evaluation Metrics

Tokenization Accuracy:

- Precision, Recall, and F1 Score ensure the robustness of preprocessing.

ROUGE Scores:

- Evaluates summarization quality and content retention.

Integration with Streamlit

Choice of Interface:

- Sidebar for task selection.
- Input boxes and interactive buttons for summarization and evaluation.

Neural Network Model

(Not included in the current project but can be added for advanced summarization tasks)

Architecture: Use a sequence-to-sequence model with an encoder-decoder framework.
Optimizer: Adam optimizer for better learning.
Evaluation: Use BLEU scores or further enhance with fine-tuned ROUGE metrics.

EVALUATION:

The evaluation for the Summarization Framework focuses on measuring both preprocessing accuracy and the quality of the generated summaries. Below are the performance metrics used to assess the project:

1. Tokenization Accuracy

This metric evaluates the precision of text tokenization, which is a critical preprocessing step. The model compares the tokenization output with a reference tokenizer and measures:

Precision:

- Percentage of correctly identified tokens.

Recall:

- Percentage of reference tokens correctly identified

F1:

- Score: Harmonic mean of Precision and Recall.

2. Summarization Quality

Generated summaries are assessed for coherence and relevance using the following metrics:

ROUGE Scores:

- ROUGE-1: Measures overlap of unigrams (individual words).
- ROUGE-2: Measures overlap of bigrams (pairs of consecutive words).
- ROUGE-L: Measures the longest common subsequence between the generated and reference summaries.

3. System Performance

- Response Time: Average time taken to generate a summary for a given input text.
- Scalability: Number of summarization requests handled efficiently per unit time for varying text lengths.

RESULT

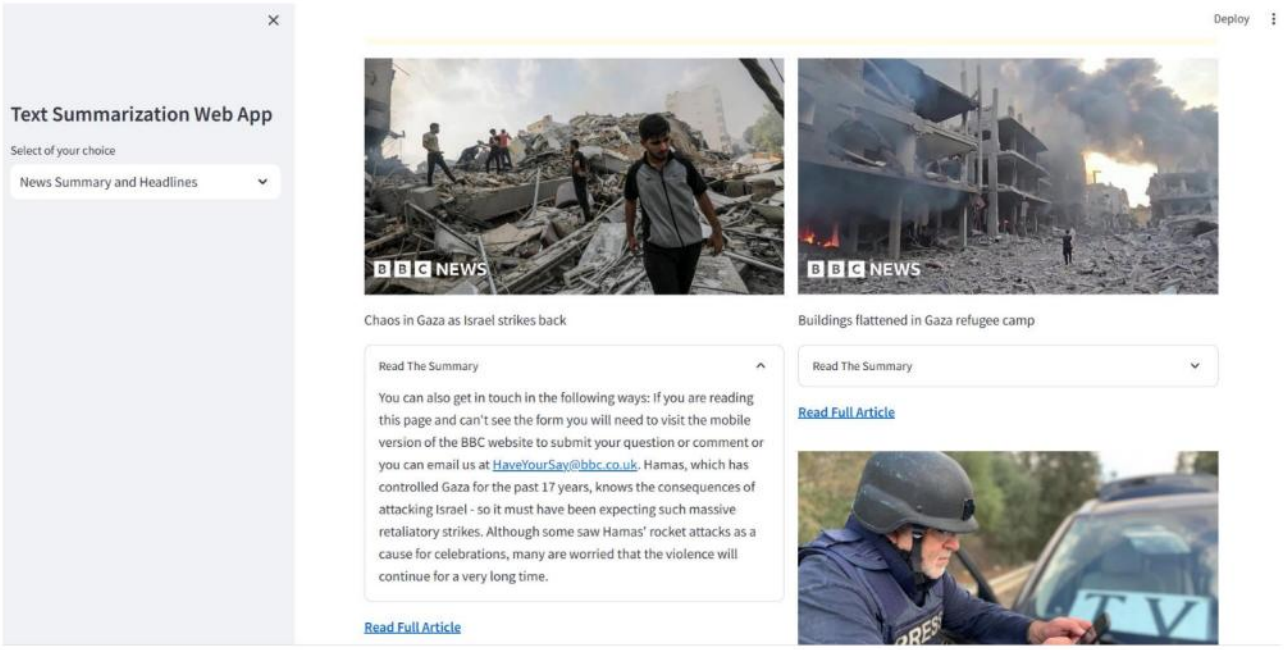

Fig: News Headlines and Summary

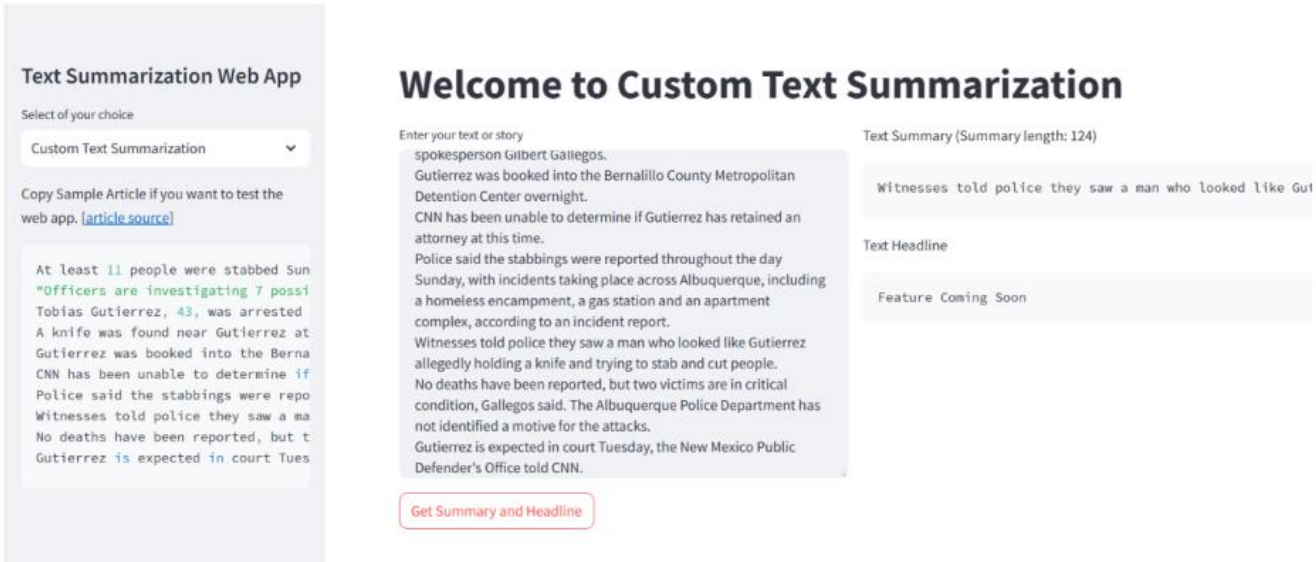

Fig: Custom Text Summarization

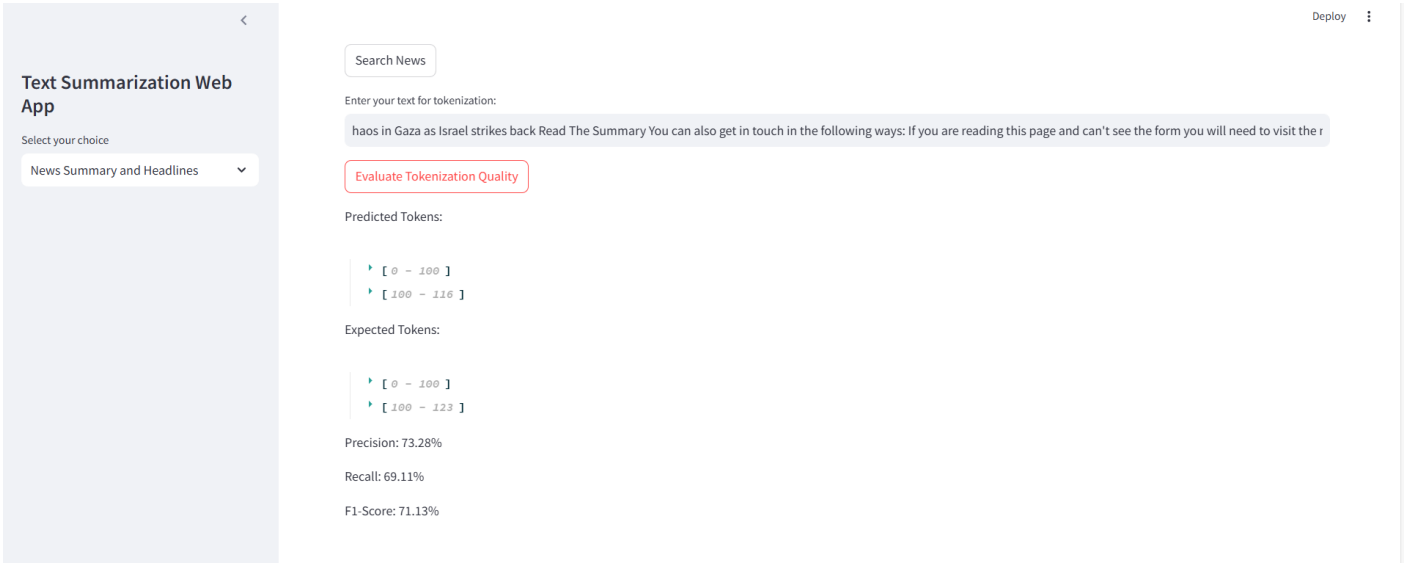

Fig: Tokenization Quality Scores (Precision, Recall and F1-Score)

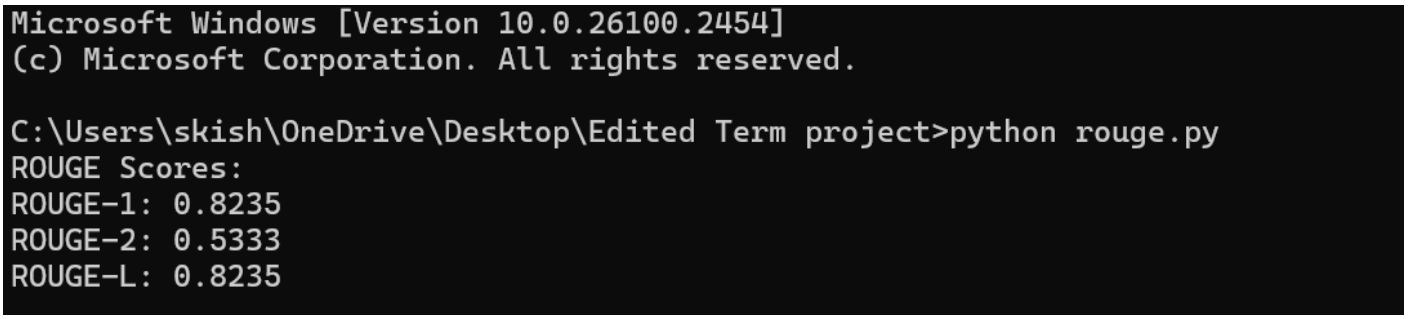

Fig: Rouge Scores (Rouge 1,2 and L)

Conclusion:

The Summarization Framework successfully achieves its objective of generating efficient and coherent summaries for custom text and news articles by leveraging advanced NLP techniques, ensuring accurate text preprocessing and robust summary generation. The integration of tokenization accuracy and ROUGE scores enhances the model's reliability, while a user-friendly Stream lit interface provides an interactive platform for summarization and evaluation. Scalability and performance optimization enable the framework to handle large datasets and multiple requests, making it suitable for real-world applications. Additionally, the Thunder Client Python library simplifies API testing with its lightweight interface, and the Stream lit library facilitates the creation of interactive web applications for data science and machine learning projects. Text summarization using NLP offers significant potential in extracting key information from large text volumes across various domains such as news, research, and social media analysis. Advancements in NLP techniques and powerful libraries make text summarization increasingly accessible, efficient, and applicable, laying a strong foundation for building more versatile system.