TEXT SUMMARIZER

Introduction

In the digital age, the exponential growth of textual data necessitates efficient methods to condense information without losing its essence. Text summarization addresses this need by producing concise versions of longer documents.

This project focuses on two primary approaches:

- **Extractive summarization**: Selects and compiles existing sentences from the source text.
- **Abstractive summarization**: Generates novel sentences that encapsulate the main ideas.

Dataset Description and Features

Dataset link : https://www.kaggle.com/datasets/pariza/bbc-news-summary

The **BBC News Summary dataset** comprises **2,225 news articles** sourced from the BBC website, spanning five topical categories:

- Business
- Entertainment
- Politics
- Sport
- Tech

Each article is paired with a human-written summary, facilitating supervised learning for summarization tasks.
The dataset includes:

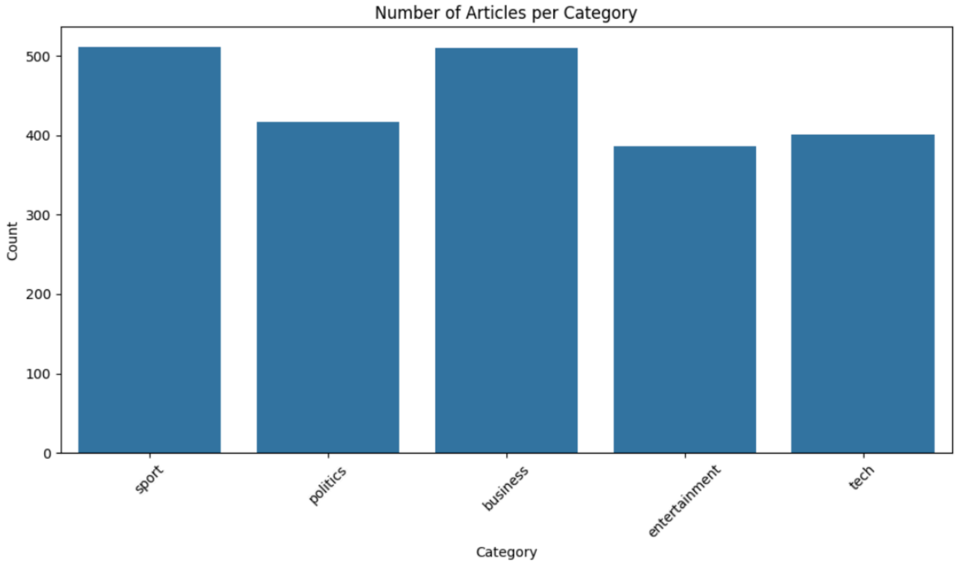
- Title
- Full article text
- Category
- Corresponding summary

Exploratory Data Analysis (EDA)

1. Distribution of Articles by Category

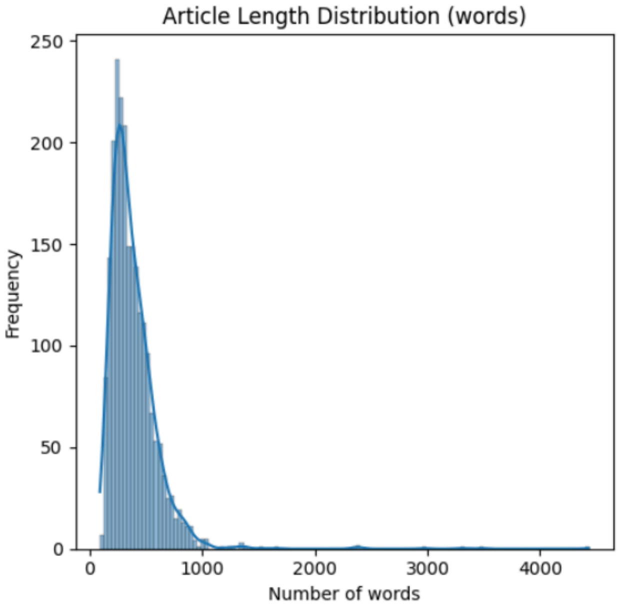
The dataset consists of **2,225** articles distributed across five categories:

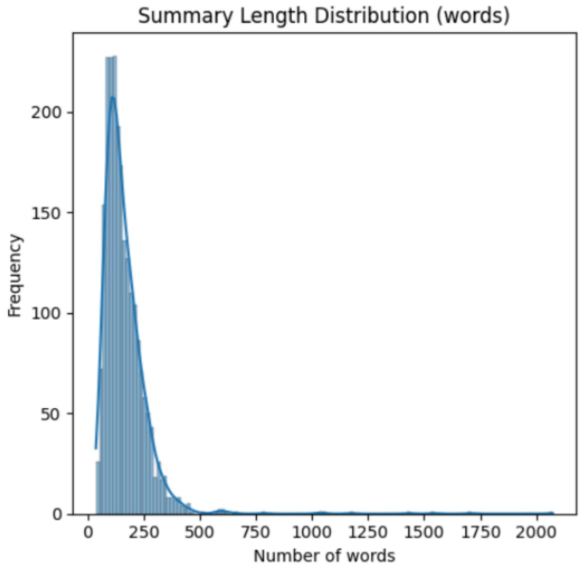
- **Sport:** 511 articles
- **Politics:** 417 articles
- **Business:** 510 articles
- **Entertainment:** 386 articles
- **Tech:** 401 articles



2. Article Length Statistics

- Mean article length: 384.04 words
- Median article length: 332 words
- Minimum article length: 89 words
- Maximum article length: 4,432 words



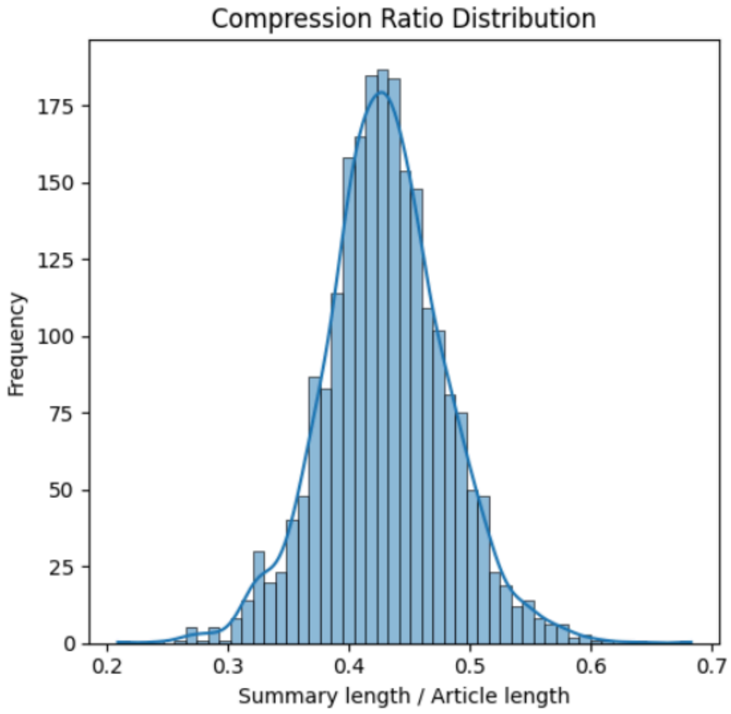


3. Summary Length Statistics

- Mean summary length: 165.17 words
- Median summary length: 142 words
- Minimum summary length: 38 words
- Maximum summary length: 2,073 words

4. Compression Ratio Statistics

- Mean compression ratio: 0.4302
- Median compression ratio: 0.4292



Preprocessing Techniques Used

To prepare the data for summarization:

1. Text Cleaning

- Lowercasing of the characters in the sentences
- Removal of newline characters and extra whitespace.
- Standardization of abbreviations.

2. Sentence Segmentation

- Splitting articles into individual sentences with careful punctuation handling.

3. Tokenization

- Tokenizing text using the **BART tokenizer** for abstractive summarization.

4. Dataset Splitting

- Dividing the dataset into **training**, **validation**, and **test** sets using stratified sampling.

Algorithms/Techniques/Models Used

1. Extractive Summarization

- **TF-IDF Vectorization:** Converts sentences into numerical vectors based on term importance.

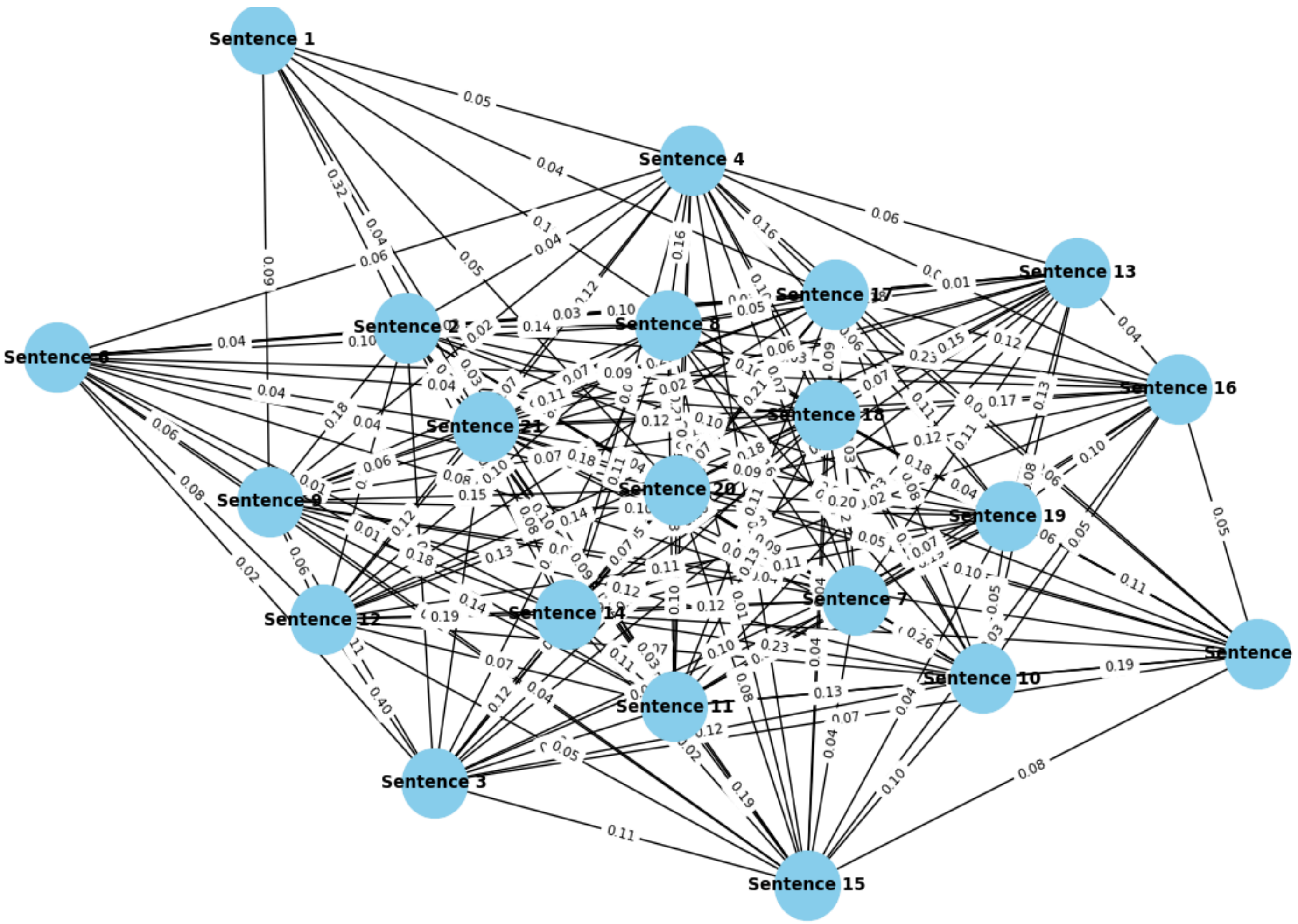
- **Cosine Similarity:** Measures similarity between sentence pairs to form a similarity matrix.

- **Similarity Matrix Construction:** Builds a weighted matrix representing sentence connections.

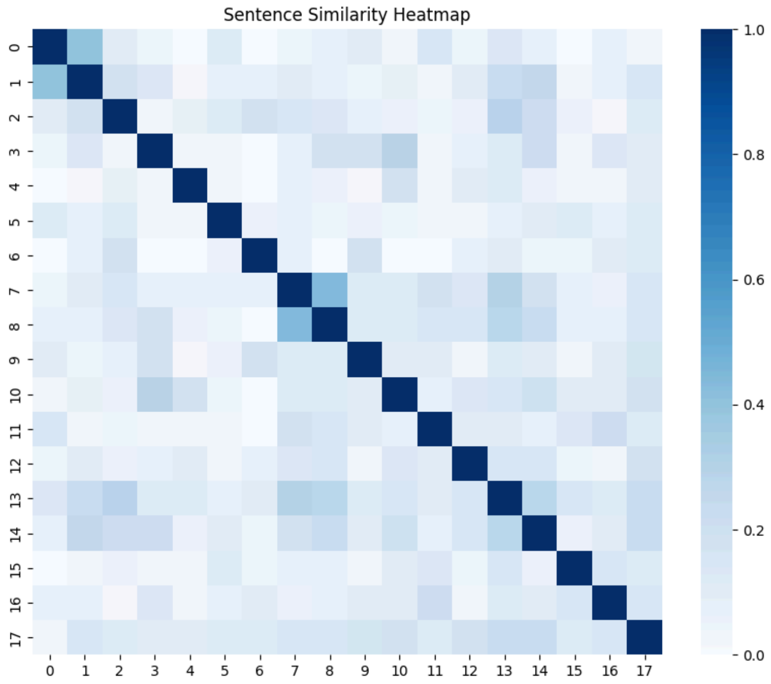
- **Sentence Graph Formation:** Creates a graph where sentences are nodes and edges are weighted by similarity.

- **PageRank Algorithm:** Ranks sentences based on centrality, selecting the most important ones for the summary

Sentence graph :


Sentence 1
Sentence 4
Sentence 13
Sentence 17
Sentence 8
Sentence 2
Sentence 6
Sentence 16
Sentence 18
Sentence 21
Sentence 20
Sentence 9
Sentence 19
Sentence 7
Sentence 12
Sentence 14
Sentence
Sentence 10
Sentence 11
Sentence 3
Sentence 15
0.05
0.04
0.06
0.32
0.04
0.05
0.09
0.16
0.06
0.04
0.10
0.06
0.08
0.02
0.06
0.40
0.11
0.19
0.08
0.10
0.05
0.19
0.11
0.04

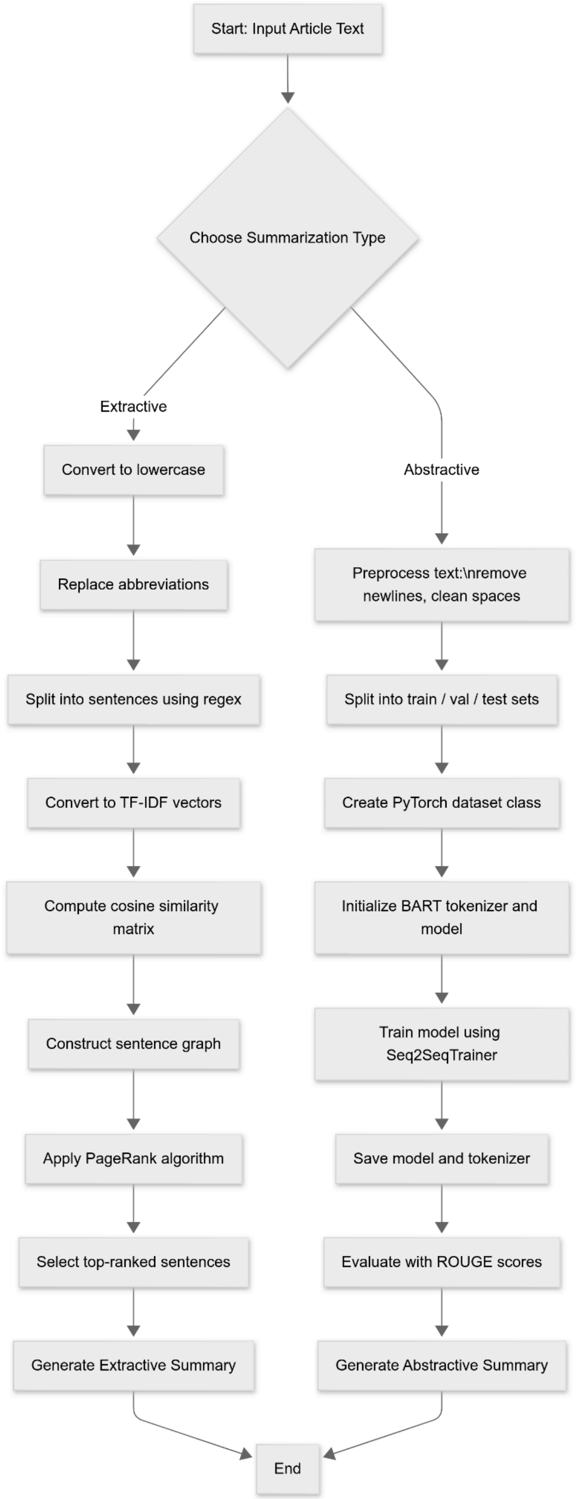
Heatmap showing similarity between sentences :



2. Abstractive Summarization

- **Model Used**: `facebook/bart-base` from Hugging Face
- **Fine-Tuning**: On BBC News Summary dataset with custom hyperparameters
- **Generation Parameters**:
  - Max summary length
  - Beam search (e.g., `num_beams=4`)
  - Length penalty to control verbosity

Overall Flow :


Start: Input Article Text
Choose Summarization Type
Extractive
Abstractive
Convert to lowercase
Replace abbreviations
Split into sentences using regex
Convert to TF-IDF vectors
Compute cosine similarity matrix
Construct sentence graph
Apply PageRank algorithm
Select top-ranked sentences
Generate Extractive Summary
Preprocess text:\nremove newlines, clean spaces
Split into train / val / test sets
Create PyTorch dataset class
Initialize BART tokenizer and model
Train model using Seq2SeqTrainer
Save model and tokenizer
Evaluate with ROUGE scores
Generate Abstractive Summary
End

Evaluation Metrics

Used **ROUGE (Recall-Oriented Understudy for Gisting Evaluation)** metrics:

- **ROUGE-1**: Unigram (single word) overlap
- **ROUGE-2**: Bigram (two-word) overlap
- **ROUGE-L**: Longest Common Subsequence (LCS)

Conclusion

In this project, we developed a comprehensive text summarization system that integrates both extractive and abstractive techniques to generate concise and meaningful summaries. The extractive approach ensured that key sentences were preserved directly from the text, maintaining factual accuracy, while the abstractive method to rephrase and condense information in a more human-like manner. By combining these approaches, our model effectively balances informativeness and readability.

Through rigorous evaluation, we observed that our system performs well in capturing essential details while reducing redundancy. Future improvements can include fine-tuning the abstractive model with larger datasets, enhancing coherence with advanced language models, and integrating user preferences for adaptive summarization. Overall, this project highlights the potential of hybrid text summarization in automating information processing and improving content accessibility.

Team :

- Sayali Khedkar 612203092
- Jay Kolhe 612203096
- Tejas Kolhe 612203097
- Varad Kothekar 612203099