



Text Summarization | Extractive| BLEU

Team Members:-
Lohithkumar Bollampally
Tanoj Kumar Anapana
Lavanya Gurram



Contents

Problem Statement

Methods

- Deep Dive Into Extractive Text Summarization Methods
- Graph Based Summarization Understanding

Initialization

EDA

Preprocessing

- Sentence Tokenization
- Spell Correction
- Sentence Similarity

Summarization

Conclusion

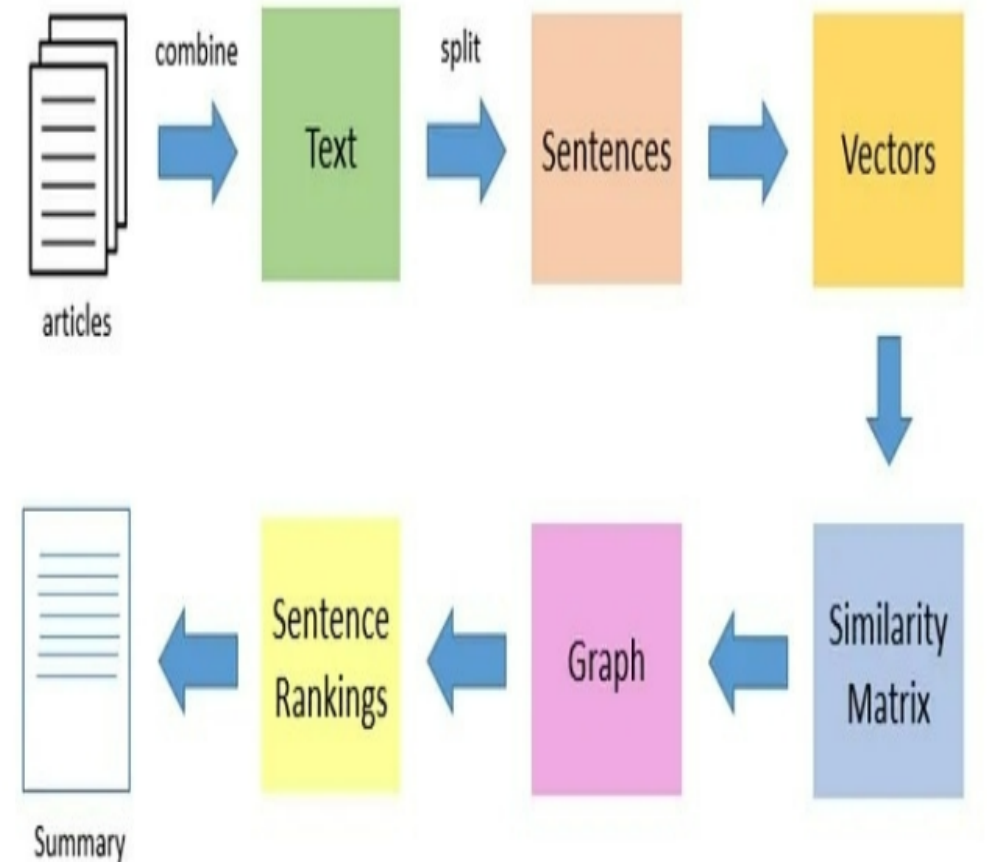
Introduction


- Text summarization is the process of generating a concise version of a text that captures its essential details. While humans naturally grasp the main ideas by reading, machines achieve this through **Natural Language Processing (NLP)** techniques.
- **Key Uses of Automatic Text Summarization:**
 - Streamlining customer feedback into brief reviews.
 - Producing highlights of news articles.
 - Summarizing meeting discussions into actionable reports.
- This Notebook delves deeper into the fascinating application of NLP for automating the summarization process.




APPROACH - ALGORITHMS, DATASETS, MODELS, TOOLS, AND TECHNIQUES

- Algorithms: TextRank for extractive summarization, with potential benchmarking against LexRank or BERT-based models. Datasets: We took data set from the Kaggle.
- Models and Tools: Python and NLP Libraries: Using libraries like NLTK, SpaCy, or Gensim for text processing and TextRank implementation.
- Evaluation Tools: ROUGE and BLEU metrics to assess summary accuracy and coherence.
- Techniques: Preprocessing (tokenization, stop word removal), sentence similarity calculation, and graph construction for TextRank.





Techniques for Text Summarization

- 
- **Extractive Summarization**
 - Picks key sentences directly from the original text.
 - Simple and quick to implement.
 - Often uses unsupervised methods that don't need training.
 - **Abstractive Summarization**
 - Understands the main ideas of the text.
 - Creates a new summary in its own words.
 - More complex, as it requires generating natural language

Extractive Summarization Explained

- **How it Works:**

It picks sentences directly from the document to create a summary. These sentences appear exactly as they do in the original text.

- **Key Challenge:**

Deciding which sentences are important enough to include in the summary.

- **Solution:**

- Each sentence is scored based on certain features (e.g., length, keywords, position).
- Sentences are ranked by their scores.
- The top-ranked sentences are included in the summary



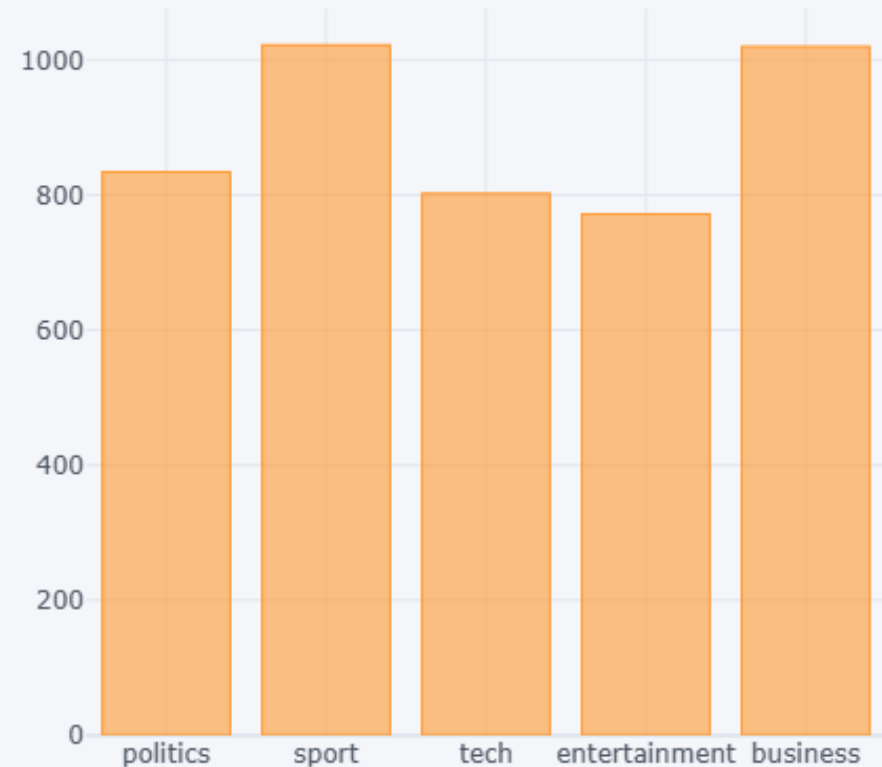
Graph Based Summarization Understanding

- The goal of this method is to pick the most important sentences from a group of sentences by looking at how similar they are to each other. The idea is that important sentences will be more connected to others.
- **Cosine Similarity** is used to measure how similar two sentences are. It gives a score between 0 and 1, where 1 means the sentences are very similar.
- **TextRank Algorithm** works like a system for ranking sentences based on their importance. It's inspired by the **PageRank Algorithm**, which Google uses to decide the importance of web pages.
- In **PageRank**, web pages are like points (nodes) in a network, and the links between them are connections (edges). The algorithm scores each page based on how likely someone is to visit it. Similarly, in TextRank, sentences are like nodes, and their similarity scores are like edges. The algorithm ranks sentences to find the most important ones



Distribution of Number of Articles in Each Category

- This EDA step involves analyzing how many articles exist in each category (e.g., "Sports," "Business," "Technology"). By visualizing this distribution, we can identify:
 - **Which categories are more heavily represented** (e.g., one category might have many more articles than others).
 - **Potential data imbalances**, which could affect the model's performance if a category is underrepresented.

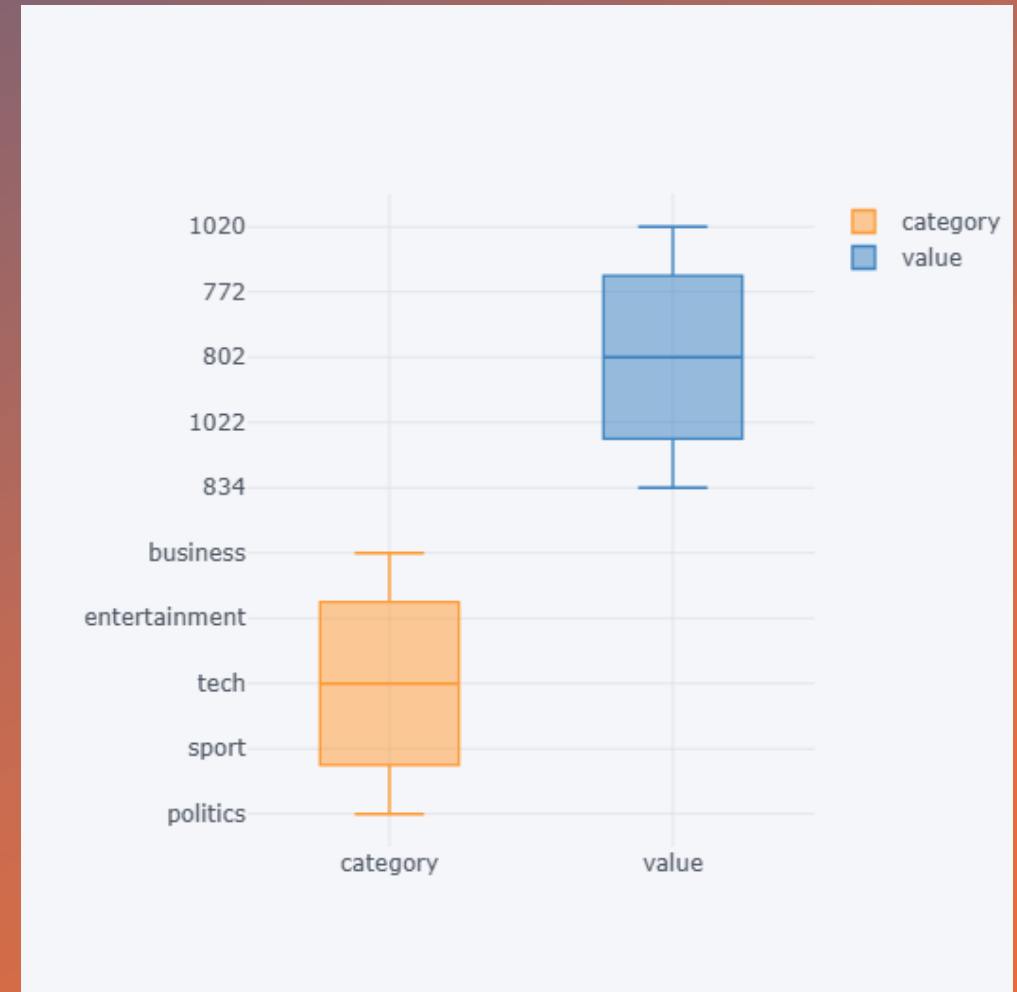


Distribution of Category and its Values

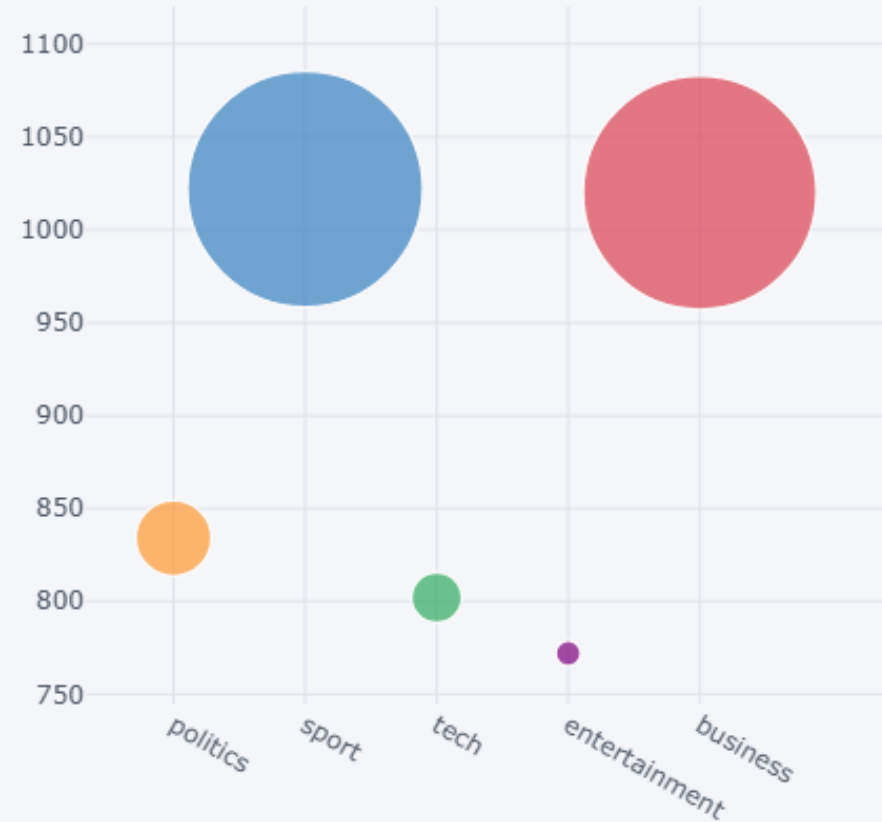
This analysis focuses on the values (or counts) within each category. This could refer to the different subcategories, such as:

How many articles fall under each subcategory within a major category (e.g., "Technology" might have "Gadgets," "AI," "Software").

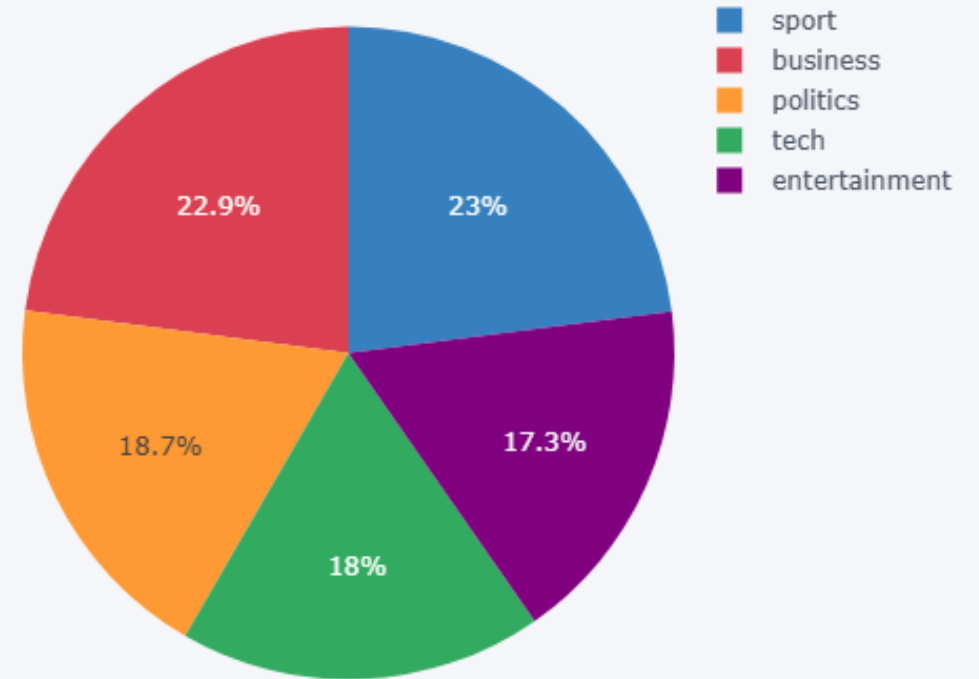
Visualization of value distributions across categories (e.g., bar charts or box plots) can help us understand if the distribution is skewed or balanced



- **3. Distribution Size of Each Category**
- This EDA step looks at the **size** of articles in each category, typically measured by:
- **Word count, character count, or article length.**
- Understanding the size distribution can help us understand if some categories tend to have longer or shorter articles.



- **4. Coverage Ratio of Each Category**
- The **coverage ratio** refers to how well each category is represented in the dataset, and it can be calculated as:
- **$(\text{Number of articles in a category}) / (\text{Total number of articles})$.**
- This helps us understand the **proportion of the dataset** dedicated to each category, and whether any categories are over or underrepresented



preprocessing



SENTENCE
TOKENIZATION



SPELL CORRECTION



TOKEN SIMILARITY

Summarization

- Read the Article and Extract Text:**

- First, we load the article and extract the actual text content (removing any unnecessary information like metadata).

- Generate Sentence Tokens:**

- Next, we split the text into sentences, so that we can work with each sentence individually.

- Compute Cosine Similarity:**

- For each sentence, we calculate how similar it is to the others using **cosine similarity**. This helps us understand how related the sentences are to each other.

- Build a Graph of Sentence Similarities:**

- Using **NetworkX**, we create a graph where each sentence is a node, and edges represent how similar two sentences are (based on the cosine similarity).

- Rank Sentences with PageRank Algorithm:**

- We apply the **PageRank algorithm** (the same idea used by Google to rank websites) to the graph. Sentences that are more important or central in the network will get higher ranks.

- Collect the Top N Sentences:**

- Finally, we pick the top N ranked sentences (the most important ones) and use them to create a summary of the entire article.

Conclusion



Data Collection: Collected BBC articles and their summaries for reference and comparison.



Text Summarization Methods: Explored both **Extractive** and **Abstractive** summarization techniques.



Focus on Extractive Summarization: Deep dive into **Extractive methods** for selecting key sentences.



Graph-Based Approach: Used a graph-based method for summarization:

Converted articles into sentences.

Calculated sentence similarities to build a graph.

Applied **PageRank algorithm** to rank sentences.

Selected the top-ranked sentences to create a summary.



Validation: Used **Similarity Score** for evaluation, finding it more reliable than **BLEU Score** for our case.