Project Report : CS 5660

Yash Sohagia
California State University, Los Angeles
ysohagi@calstatela.edu

Sarthak Shah
California State University, Los Angeles
sshah77@calstatela.edu

Darshan Basavarajappa
California State University, Los Angeles
dbasava@calstatela.edu

Vyoma Sadhu
California State University, Los Angeles
vsadhu@calstatala.edu

Ju Zi Hein
California State University, Los Angeles
jhein4@calstatela.edu

Abstract

Automatic text summarization is basically summarizing of the given paragraph using natural language processing and machine learning. There has been an explosion in the amount of text data from a variety of sources. This volume of text is an invaluable source of information and knowledge which needs to be effectively summarized to be useful. In this review, there are two primary methods utilized in automatic text summarization: extractive and abstractive summarization. These methods employ distinct approaches to enhance the effectiveness of text summarization techniques. Python provides numerous packages and methods for implementing text summarization using natural language processing (NLP). In this paper, a model is developed using LexRank and TextRank in NLP to achieve improved results in text summarization. This approach enhances the overall coherence of the summary, enabling readers to comprehend the essence of the text more effectively.

1. Introduction/Background/Motivation

The core purpose of automatic text summarization is to extract essential and concise information from vast amounts of data. As the internet continues to grow exponentially, retrieving the key data becomes increasingly challenging and time-consuming. Automatic text summarization alleviates this burden by enabling users to efficiently gather vital information from extensive sources. Graph-based ranking algorithms like TextRank, Hyperlinked Induced Topic Search, and Positional Power Function are widely employed in this field. In this paper, we focus on implementing the TextRank and LexRank algorithms, which facilitate rapid and effortless information retrieval.

There are two prominent types of summarization algorithms:

Extractive summarization systems: These systems generate summaries by selecting and combining important sentences or parts of the source text. Sentence importance is determined based on linguistic and statistical features. Abstractive summarization systems: These systems create summaries by generating new phrases, potentially rephrasing or incorporating words not present in the original text. Abstractive approaches are more challenging as they require the model to comprehend the document and express that understanding using concise and coherent language. While abstractive summarization holds the promise of grammatically correct and cohesive summaries, it often struggles with summarizing lengthy and complex texts due to their restrictive nature. By exploring and implementing the TextRank and LexRank algorithms, this research aims to enhance the efficiency and accuracy of text summarization, facilitating easy access to relevant information from large volumes of data.

Objectives: The objective of the project is to understand the concepts of natural language processing and creating a tool for text summarization. The concern in automatic summarization is increasing broadly so the manual work is removed. The project concentrates creating a tool which automatically summarizes the document.

Dataset : In our project, we will use BBC News Article dataset for extractive text summarization which has four

hundred and seventeen political news articles of BBC from 2004 to 2005 in the News Articles folder. For each articles, five summaries are provided in the Summaries folder. The first clause of the text of articles is the respective title.

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

2. Approach

The project is wide in scope, all of the limitations may seem to contradict that, but they are the only restrictions applied. This project looks at both single document summarization by using textrank and multi document summarization by using lexrank. Also, the summaries produced are largely extracts of the document being summarized, rather than newly generated abstracts. The parameters used are optimal for news articles, although that can be changed easily. In our busy schedule, it is very difficult for us to go through the entire article or document. So, we prefer to read summary. In this paper we are going to summarize the large text into a short summary which reduces reading time for users.When developing text summarization using NLP, several challenges and problems such as Out-of-Vocabulary Words, information loss and ambiguity also arise. However, we addressed these challenges by a combination of advanced NLP techniques, domain-specific training data, careful feature engineering, and continuous model refinement to improve the accuracy, coherence, and fluency of the generated summaries. Code Repositories and Sources: We utilized various code repositories and sources for reference and implementation guidance. Specifically, we referred to the following resources: Repository :

3. Experiments and Results

NLP is a part of Artificial Intelligence reasoning that manages analyzing,understanding, finding and producing the dialects that people use in a characteristic manner to make interface with PCs in both composed and spoken settings utilizingcommon human dialects rather than codes.

	path	filename	category	article_or_summary
0	C:\Users\ashah77\Desktop\5660 project\BBC News...	001.txt	business	News Articles
1	C:\Users\ashah77\Desktop\5660 project\BBC News...	002.txt	business	News Articles
2	C:\Users\ashah77\Desktop\5660 project\BBC News...	003.txt	business	News Articles
3	C:\Users\ashah77\Desktop\5660 project\BBC News...	004.txt	business	News Articles
4	C:\Users\ashah77\Desktop\5660 project\BBC News...	005.txt	business	News Articles
...				
8895	C:\Users\ashah77\Desktop\5660 project\BBC News...	397.txt	tech	Summaries
8896	C:\Users\ashah77\Desktop\5660 project\BBC News...	398.txt	tech	Summaries
8897	C:\Users\ashah77\Desktop\5660 project\BBC News...	399.txt	tech	Summaries
8898	C:\Users\ashah77\Desktop\5660 project\BBC News...	400.txt	tech	Summaries
8899	C:\Users\ashah77\Desktop\5660 project\BBC News...	401.txt	tech	Summaries

8900 rows × 4 columns

Figure 1. Dataset Example

LexRank Algorithm

LexRank is a graph-based unsupervised method for text summarization that utilizes sentence similarity and centrality scoring. The core concept behind LexRank is the notion of sentence recommendations. When one sentence is highly similar to numerous others, it is considered important and likely to be included in the summary. The importance of a sentence is not only based on its own characteristics but also on the importance of the sentences that recommend it. Therefore, to receive a high ranking and be selected for the summary, a sentence must exhibit similarity to many other sentences that, in turn, share similarities with a broad range of additional sentences. This intuitive approach enables the algorithm to be applied to any given text, regardless of its content.

The following depicts the steps of summarizing texts using LexRank:

Data Loading and Analysis: The code loads the BBC News Summary dataset from a specific directory using os.walk. It extracts information like file paths, filenames, categories, and whether it's an article or summary and creates a Pandas DataFrame for analysis.The code utilizes Plotly Express and Cufflinks to visualize the distribution of categories in the dataset.

Text Preprocessing:

The code imports necessary libraries such as NLTK for text preprocessing tasks.It defines a function read article that tokenizes the text into sentences using NLTK's sent tokenize.The code applies sentence tokenization to an example article from the dataset.

Text Correction:

The code uses the TextBlob library to perform text correction on the sentences extracted from the article.It iterates over each sentence, applies text correction using TextBlob's correct() function, and stores the corrected sentences.

Sentence Similarity: The code uses the Universal Sentence Encoder from TensorFlow Hub to calculate the similarity between two sentences.It demonstrates sentence similarity by computing the similarity score between two sample sentences.

Text Summarization:

The code defines a function build similarity matrix that calculates the similarity matrix between sentences based on sentence similarity scores.It builds a graph representation using NetworkX based on the similarity matrix and visualizes it using Bokeh.The code generates a summary by ranking sentences using PageRank on the similarity graph and selecting the top N sentences.It compares the generated summary with the actual summary provided in the dataset and computes BLEU score and sentence similarity score.

4. TextRank Algorithm

Text rank algorithm is a diagram-based positioning model for text processing which can be used in order to find the most applicable sentences in text and also to find keywords. Text rank algorithm is similar to the page rank algorithm. Page rank algorithm is used to mark Web Pages in web search conclusions and in web usage mining. In text rank algorithm, the position of Webpages sentences are

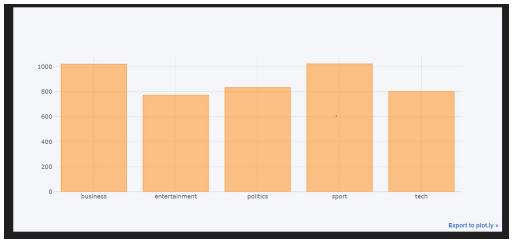

Figure 2. Barchart of news articles

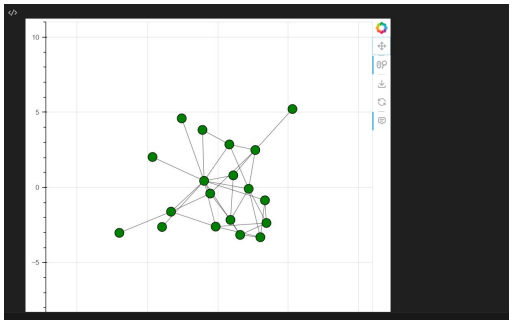

Figure 3. Network graph of news articles

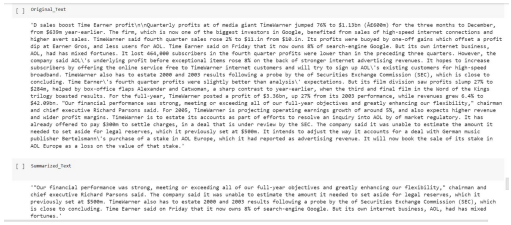

Figure 4. Summarizing text paragraph using LexRank

taken. 1. Identify content units that best characterize the current task, and add them as vertices in the diagram. 2. Identify the relations that append the content units, and in the chart utilizing these relations draw edges between vertices. Edges can be unweighted or weighted and undirected or coordinated. 3. And at that point loop the diagram-based positioning algorithm until union. 4. Based on their last score, they mastermind the vertices. For positioning and determination choices Use the qualities appended to every vertex. 5. At last, the highest-level sentences will shape a synopsis. In the code, the article is tokenized into sentences, and a sentence similarity matrix is computed using cosine similarity. The sentences are then ranked based on their scores, and the top N sentences are selected as the summary. TextRank algorithm used in the code snippet incorporates natural language processing (NLP) techniques. Here's how NLP is involved in the process: Tokenization: The article is tokenized into sentences using the sent tokenize function from the nltk.tokenize module. Tokenization is the process of splitting text into individual units, such as sentences or words, to facilitate further analysis. Similarity Calculation: The algorithm uses NLP techniques to calculate the similarity between sentences. It employs the CountVectorizer class from the sklearn.feature extraction.text module to convert the sentences into numerical vectors. Then, cosine similarity is computed using cosine similarity from sklearn.metrics.pairwise to measure the similarity between sentence vectors.

Sentence Ranking: The sentences are ranked based on their similarity scores. The higher the score, the more important the sentence is considered to be. This ranking process involves NLP techniques for vectorization, similarity calculation, and sorting. Summary Generation: The top-ranked sentences are selected as the summary. The summary generation step relies on NLP techniques for sentence selection and text concatenation. Although TextRank is primarily an extractive summarization method, it utilizes NLP techniques like tokenization, vectorization, and similarity calculation to analyze and process the text data effectively.

Figure 5. Summarizing text paragraph using TextRank

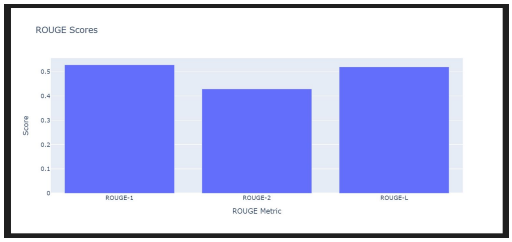

Figure 6. ROUGE metric using textrank

Naive Bayes Algorithm

Naive Bayes Algorithm Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other. The dataset is divided into two parts, namely, feature matrix and the response/target vector. The Feture matrix (X) contains all the vectors(rows) of the dataset in which each vector consists of the value of dependent features. The number of features is d i.e. X = (x1,x2,x2, xd). The Response/target vector (y) contains the value of class/group variable for each row of feature matrix. Naive Bayes assumes that each feature/variable of the same class makes an: independent, equal contribution to the outcome.

Figure 7. Word map using bayes naive

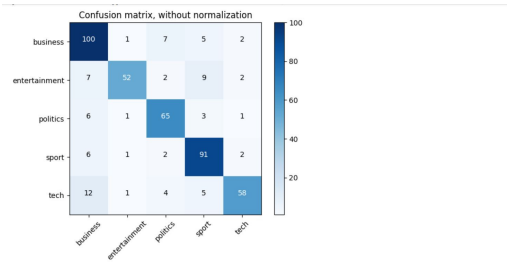

Figure 8. Confusion metric

Best hyper parameters with vectorizer	Accuracy
TextRank	53%
LexRank	56%
Naive Bayes	82%

Table 1. Results of each Vectorizer method with best hyper parameters found with GridSearchCV.

5. Future Scope

In the future,advancements in TextRank and LexRank can focus on improving their performance, adapting them to specific domains, handling multiple documents, incorporating user queries, enabling real-time summarization, supporting multiple languages, and enhancing user interactivity. These developments can significantly enhance the utility and effectiveness of text summarization using these algorithms.

Conclusion

The paper demonstrates that we use advanced techniques to apply on the document for text summarization using an extractive summarization method called TextRank algorithm and unsupervised graph based LexRank algorithm. At first, we loaded necessary libraries and related functions in python and then code was implemented to summarize the text. Afterwards, a model is proposed with slight expansions to improve by showing the outline text. The techniques that are presented in this paper to get better result in a text summarization library in NLP. With this the overall meaning of the document can be understood easily.

6. Work Division

Student Name	Contributed Aspects	Details
Sarthak Shah	TextRank , naive bayes	Contributed in the implementation and testing of TextRank model. Helped in improving accuracy and implementation of naive bayes algorithm and model
DarshanBasavarajappa	LexRank	Contributed in the implementation LexRank model. Helped in improving accuracy and project report.
Vyoma Sadhu	LexRank	Contributed in the implementation and testing of LextRank model. Helped in improving accuracy.
Ju Zi Hein	TextRank	Contributed in the development of TextRank model.Helped in improving accuracy and updating project report.
Yash Sohagia	TextRank , naive bayes	Contributed in the development of TextRank model.Helped in improving accuracy and updating project report . and implementation of naive bayes algorithm and model

Table 2. Contributions of team members.