BBC text summarization project

Mohamed,Shahd.Ahmed,Esraa.
Ali,Enas.Elmekawy,Hassnaa.
Department of Mechanical Engineering
university of u ottawa
Ottawa, Ontario, Canada
loansandawards@uOttawa.ca

***Abstract*— Text summarization involves condensing important information from a text into a summary. Its significance has grown recently, especially in the news, where a concise summary is crucial. Automatic text summarization is a well-established task in Natural Language Processing (NLP). There are two main approaches: Extractive summarization, which selects key sentences or phrases directly from the original text, and Abstractive summarization, which comprehends the source text and generates new sentences to enhance the summary's coherence and meaning.**

The goal of this study is to implement and compare various automatic summarization methods to understand their implementation time, accuracy, and human-like quality of generated summaries. This will involve summary scoring and manual inspection to evaluate the strengths and weaknesses of each technique.

Keywords—Text summarization, Automatic text summarization, Extractive summarization, Abstractive summarization, Natural Language Processing (NLP), Summary scoring, Implementation time, Accuracy, Human-like quality.

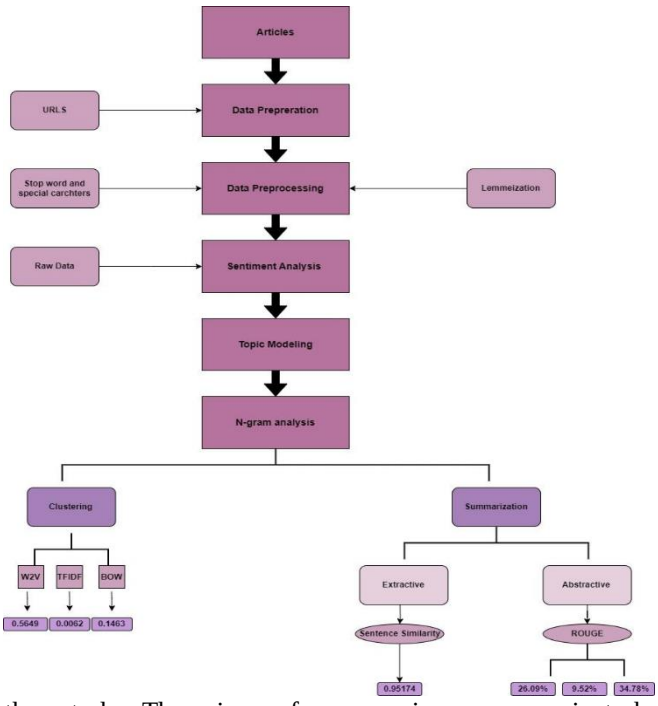
I. INTRODUCTION

In recent times, there has been a significant increase in the volume of text data available from various sources. This abundance of information contains valuable expertise and facts that need to be efficiently summarized to be practical and useful. The main objective is to automate the process of text summarization, as people are finding it overwhelming to deal with the vast amount of knowledge and documents available online due to the exponential growth of the Internet. As a result, there is a growing necessity for extensive research and development in the field of automated text summarization. The study of text summarization has become increasingly common as the internet continues to expand with an ever-growing wealth of knowledge.

Text summarization can be achieved through two main approaches: Extractive (EXT) and Abstractive (ABS) summarization. EXT relies on using the same sentences present in the document to create a summary, while ABS takes a more general approach by focusing on the key concepts of the text. Single-document summarization techniques generate summaries for individual documents, while multi-document summarization deals with summarizing multiple documents at once.

Text summarization finds extensive applications in various fields such as news, medicine, law, engineering, etc. For instance, summarizing lengthy news articles enables readers to grasp information on diverse topics quickly.

This paper aims to examine various text summarization techniques that have been employed. It also explores some concepts related to NLP tasks, such as topic modeling and sentiment analysis, as well as utilizing unsupervised machine learning techniques like clustering to gain insights through all



these tasks. The primary focus remains on our main task, which is text summarization.

II. SYSTEM ARCHITECTURE

Text summarization is a crucial and valuable Natural Language Processing (NLP) project that addresses the challenge of condensing lengthy documents or articles into shorter, coherent, and informative summaries. It plays a significant role in various applications and domains due to its numerous benefits. In today's digital age, we are inundated with vast amounts of information. Text summarization helps users quickly grasp the main points of a document without having to read the entire content, saving time and effort. Summaries allow users to get the gist of multiple documents, articles, or news stories in a short amount of time. It is particularly useful when dealing with time-sensitive or high-volume data. Also, summarization makes complex topics more accessible to a broader audience. It can benefit people with limited reading capabilities, cognitive disabilities, or language barriers, enabling them to comprehend important information more easily. Summarization is an active area of research within the field of NLP. Advancements in summarization algorithms contribute to improving other NLP tasks like question-answering systems, dialogue systems, and machine translation.

Text summarization encompasses two primary approaches: extractive (selecting and assembling important sentences from the original text) and abstractive (generating new sentences to form the summary). Both methods have their unique advantages and use cases.

DATASET

To address these approaches, we used the Extreme Summarization (XSum) dataset which serves as a valuable resource for evaluating abstractive single-document summarization systems. Its primary objective is to generate concise one-sentence summaries that answer the question, "What is the article about?". This dataset comprises 226,711 news articles, each accompanied by a single-sentence summary. The articles are sourced from BBC articles spanning the years 2010 to 2017, encompassing a diverse range of domains such as News, Politics, Sports, Weather, Business, Technology, Science, Health, Family, Education, Entertainment, and Arts. The official random split distributes the data into training (90%), validation (5%), and test (5%) sets, containing 204,045, 11,332, and 11,334 documents, respectively.

PREPROCESSING

Preprocessing is a critical and fundamental step in Natural Language Processing (NLP) that involves transforming raw text data into a format that can be more effectively analyzed and processed by NLP algorithms and models. The importance of preprocessing in NLP cannot be overstated, as it directly impacts the quality and accuracy of downstream NLP tasks and applications.

To begin with, we utilized the provided URLs file to extract different domains for the articles contained in the dataset using regular expressions (regex). This allowed us to categorize our data into various article domains, such as politics, sports, health, education, and more.

Our analysis focused on a subset of the data, specifically the first 3,000 articles. For this subset, we conducted various statistical calculations to gain valuable insights. The average article length in this dataset was found to be approximately 432.35 words, providing a typical representation of the article size. The range of article lengths varied from a minimum of 12 words to a maximum of 3576 words, indicating diverse article sizes. Moreover, we identified a total of 51375 unique words, reflecting the lexical richness and complexity of the content in this collection.

In addition to examining the article lengths, we also analyzed the summaries provided for each article. The statistical calculations revealed an average summary length of 21.107 words, giving us an understanding of the summarization pattern across the dataset.

Within our dataset, we observed multiple domains for the articles, with the 'news' domain being the most prevalent, accounting for 55.76% of the articles. The 'sports' domain followed with 24.09%, while 'business' and 'politics' domains contributed 6.46% and 4.20%, respectively.

In our experiment, we exclusively utilized articles written in the English language, carefully selected using the 'langid' library. The langid library is an asset in the realm of Natural Language Processing (NLP), as its primary function is to automatically identify the language of a given piece of text or document.

The primary objective of NLP preprocessing is to cleanse, standardize, and prepare the text data for further analysis, thereby enhancing the quality and accuracy of NLP tasks and applications. To ensure the text's quality, we performed several operations, including the removal of HTML tags/markups, punctuation, digits, non-word characters, and extra whitespace. Additionally, all text was converted to lowercase to maintain consistency throughout the analysis.

We further enhanced the data by eliminating stop words from the input text. Stop words are common words such as "the," "and" "is," etc., which typically do not contribute significantly to the text's meaning and are frequently removed in NLP tasks to reduce noise.

Furthermore, we employed lemmatization on the input text to convert words to their base or root form, thereby reducing inflected forms of words to a common base. This step aids in achieving a more standardized representation of the vocabulary and improves the performance of subsequent NLP tasks.

Moreover, we endeavored to explore additional NLP concepts, such as sentiment analysis, topic modeling and N-gram analysis. These techniques enable us to delve deeper into the underlying sentiments and themes within the text corpus, providing valuable insights into the articles' content and facilitating a more comprehensive analysis.

SENTIMENT ANALYSIS

Sentiment analysis, also known as opinion mining, is a Natural Language Processing (NLP) technique used to determine the sentiment or emotional tone expressed in a piece of text, such as a review, social media post, or customer feedback. The main goal of sentiment analysis is to understand whether the expressed sentiment is positive, negative, or neutral. We were able to generate a score for each article ranging from -1 to 1, where positive values indicate positive sentiment, negative values indicate negative sentiment and values close to 0 indicate neutral sentiment.

TOPIC MODELING

Topic modeling is a natural language processing (NLP) technique used to discover the hidden topics or themes present in a collection of documents. It is an unsupervised machine learning approach that groups similar words together to identify coherent patterns in text data. The primary goal of topic modeling is to uncover the underlying structure of a document corpus and reveal the main topics discussed in the text.

We were able to identify 5 topics that vary from governmental and public service matters, security and conflict-related issues, political matters and elections, Incidents involving the police and car accidents and finally sports, particularly games and leagues.

N-GRAM ANALYSIS

N-gram analysis is a fundamental technique in Natural Language Processing (NLP) used to analyze sequences of contiguous words or characters in a text. N-grams are essentially a set of co-occurring words or characters within a specified window size 'n'. The 'n' in n-grams represents the number of words or characters in each set.

In the context of N-gram analysis, trigrams represent sequences of three adjacent words within a text. The identified trigrams presented herein exhibit notably high frequencies, indicating their recurrent occurrence and

prevalence in the analyzed text data. The numerical values associated with each trigram signify the frequency of appearances, thereby offering valuable insights into the textual content and common word combinations present. Following the application of N-gram analysis, our observations revealed the occurrence of the trigram (win, free, kick)440 times in the dataset, the trigram (right, footed, shot) 286 times, the trigram (playback, supported, device) 240 times, the trigram (medium, playback, supported) also 255 times, and finally, the trigram (kick, defensive, half) was observed 188 times in the dataset. These findings contribute to a deeper understanding of the textual patterns and linguistic structures inherent in the analyzed text corpus. Overall, through diligent language selection, meticulous NLP preprocessing, and the exploration of various NLP concepts, our study sought to ensure data quality and facilitate sophisticated analysis, ultimately contributing to a robust and insightful research endeavor.

III. Models

Clustering

Text transformation:

Text transformation techniques such as Bag of Words (Bow), Term Frequency-Inverse Document Frequency (TF-IDF), and Word Embedding are typically applied as a preprocessing step before clustering text data. The reason for this is that clustering algorithms typically require numeric input data, and text data is inherently non-numeric. Text transformation techniques convert the raw text data into a numerical format that can be used as input to the clustering algorithm we used three techniques of text transformation are:

1-BOW (Back of words): It is a representation of text that describes the occurrence of words within a document involves two things: (A vocabulary of known words, A measure of the presence of known words).
2-TF-IDF: a technique to quantify words in a set of documents. We compute a score for each word to signify its importance.
3-Word Embedding (Word2Vec):
It is a vector representation of a particular word, that can capture context of a word in a document, semantic and syntactic similarity, relation with other words. Word embedding id built using Continuous Skip-Gram Model, the length of the dense vector to represent each token here is 150 matching for each word.

Clustering Models:

K-means:

is one of the simplest and most popular machine learning algorithms out there. It is an unsupervised algorithm as it doesn't use labelled data, in our case it means that no single text belongs to a class or group. It classifies a dataset into a 9 number of clusters. The Euclidean method as a distance metric, k-means++ as our method for initial cluster centers for k-mean, we apply k_Means with three transformers (back of words, TF-IDF, Word-Embedding).
Hierarchical clustering: In our clustering analysis, we employed the hierarchical clustering algorithm, an unsupervised learning method, to group similar data points into clusters based on their pairwise distances. The objective was to create a hierarchy of clusters, with higher-level clusters formed through merging or splitting lower-level clusters.
After careful analysis, we identified and utilized nine distinctive clusters to represent different topics in our dataset. These clusters were:
1. News
2. Sport
3. Business
4. Politics
5. Entertainment
6. Health
7. Science
8. Technology
9. Education

Each cluster represents a specific theme or category of data points that share similarities in content. This approach allows us to effectively organize and categorize the data based on its inherent patterns, without the need for labeled data or prior knowledge of the underlying classes.
By employing hierarchical clustering with nine distinct clusters, we gain valuable insights into the diversity of topics covered in our dataset, providing a comprehensive overview of the underlying themes prevalent in the data. This organization enables us to explore and analyze the data efficiently, gaining deeper understanding and facilitating more informed decision-making processes.

Dimensionality Reduction:

After text transformation, it is quite hard to deal with this huge number of patterns and here dimensionality reduction comes to just provide us with the most important features without any redundancy. T-SNE is good choices to use in such situations T-SNE is good in visualization. I used it with an agglomerative model.

Evaluation

Silhouette:
is a commonly used metric for evaluating the quality of clustering results. It measures the degree of similarity of a data point to its own cluster compared to other clusters. The silhouette score ranges from -1 to 1.
Silhouette with K-means:

```
Silhouette Score of K-means With BOW      :  0.1463
Silhouette Score of K-means With TFIDF    :  0.0062
Silhouette Score of K-means With Word2Vec :  0.5649
```

The highest score of K_Means is 0.5649 (K_Means with word2vec)

Silhouette with Hierarchical clustering:

```
Silhouette Score of Hierarchical clustering With BOW      :  0.2708
Silhouette Score of Hierarchical clustering With TFIDF    :  0.2957
Silhouette Score of Hierarchical clustering With Word2Vec :  0.3149
```

The highest score of Hierarchical clustering is 0.3149 (Hierarchical clustering with word2vec)

Extractive method

Extractive text summarization is a technique used to create concise summaries by directly selecting and extracting important sentences or phrases from the original text. It involves identifying the most relevant and informative content within the source document and presenting it as a summary without any modifications or rephrasing. Here are the steps we do to extract the summary:

- Calculation of Semantic Similarity: To determine how similar phrases are to one another in terms of meaning, it loads the Universal Sentence Encoder (USE) model from TensorFlow Hub and Establishes the'sentence_simi' function, which uses the USE model to calculate the semantic similarity score between two sentences.
- Construction of a Similarity Matrix: Creates a similarity matrix indicating the pairwise semantic similarity between each pair of sentences in the dataset by defining a second function named "build_similarity_matrix." It tokenizes the text data into sentences after preprocessing it which we discussed earlier. Then we Used the USE model and the 'build_similarity_matrix' function to determine the semantic similarity matrix for the given texts.
- Visualizing Semantic Connections: we Utilized the NetworkX to build a network graph based on sentences with strong semantic links (similarity scores >= 0.9). we used Bokeh to visualize the graph, with the sentences acting as nodes and the semantic ties as edges.
- Word Embeddings for Sentence Embeddings: we Downloaded GloVe word embeddings that have already been trained to represent words as vectors then, Averages the word embeddings of the sentence's words to create sentence vectors for each phrase in the dataset.

- The matrix of cosine similarity: Based on the sentence vectors of each pair of sentences, we created a square matrix that is entirely zeroed out to hold the cosine similarity scores between them.
- Ranking and PageRank Sentences: Creates a graph from the cosine similarity matrix and determines the nodes' (the sentences') PageRank rankings. According to their semantic linkages, sentences' importance is determined by PageRank. We Determined how essential each sentence is by ranking them according to their PageRank scores. The top-ranked sentence (one with the greatest PageRank score) is chosen from the list of sentences to create the summary.
- Comparison with Original Summary: lastly, we Calculated the semantic similarity score between the created summary and the original summary using the USE model. And that is the **similarity score**:

```
Senetence Similarity Score : 0.9517446048557758
```

ABSTRACTIVE METHOD

This method tries to rewrite and reformulate text from the original document, which is more like how a human does summarization.
Abstractive summarization concentrates on the most critical information in the original text and creates a new set of sentences for the summary. This technique entails identifying key pieces, interpreting the context, and re-creating them in a new way. Due to the difficulty of both extracting relevant information from a document as well as automatically generating coherent text, abstractive summarization has been considered a more complex problem than extractive summarization. The abstractive summarization method works well with deep learning models like the seq2seq model, LSTM, etc., along with popular Python packages (Spacy, NLTK, etc.) and frameworks (TensorFlow, Keras).

ROUGE Score:
ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation, so from its name, it refers to Recall and precision between candidate (Predicted Summary) and reference (Original Summary)
ROUGE scores are branched into ROUGE-1, ROUGE-2 and ROUGE-L scores.

ROUGE-1 Score:
ROUGE-1 Precision and Recall compare the similarity of uni-grams between reference and candidate summaries. By uni-grams, we simply mean each token of comparison is a single word.

ROUGE-2 Score:
ROUGE-2 Precision and Recall compare the similarity of bi-grams between reference and candidate summaries. By bi-grams, we mean each token of comparison is 2 consecutive words from the reference and candidate summaries.
ROUGE-L Score:
ROUGE-L Precision and Recall measures the Longest Common Subsequence (LCS) words between reference and candidate summaries. By LCS, we refer to word tokens that are in sequence, but not necessarily consecutive. To understand this, let us look at a convoluted:

TABLE I

	ROUGE-1 (uni-grams)	ROUGE-2 (bi-grams)	ROUGE-L (LCS)
Recall	0.44	0.12	0.33
Precision	0.29	0.08	0.21
F1-score	0.35	0.10	0.26

Together, ROUGE-1, ROUGE-2 and ROUGE-L Precision/Recall give a good representation of how well the model-generated summaries represent the golden-annotated summaries.

IV. Conclusion

The report presents a study on text summarization, with a focus on comparing different automatic summarization methods. The Extreme Summarization (XSum) dataset is utilized, comprising news articles from various domains. Preprocessing techniques involve extracting domains using regular expressions, analyzing article and summary lengths, and employing language identification and lemmatization. Sentiment analysis, topic modeling, and N-gram analysis are explored to gain deeper insights into the content. For text transformation, Bag of Words (Bow), Term Frequency-Inverse Document Frequency (TF-IDF), and Word Embedding methods are applied. Clustering algorithms, including K-Means and Hierarchical Clustering, are used to group similar data points into clusters representing distinct topics. The abstractive method involves rewriting text, while the extractive method selects and assembles key sentences. Evaluation is done using ROUGE scores to compare predicted and original summaries.

References

[1] https://paperswithcode.com/dataset/xsum

[2] https://paperswithcode.com/paper/pegasus-pre-training-with-extracted-gap

[3] https://paperswithcode.com/paper/text-summarization-with-pretrained-encoders

[4] https://paperswithcode.com/paper/big-bird-transformers-for-longer-sequences

[5] https://paperswithcode.com/paper/extractive-summarization-as-text-matching