Extractive Text Summarization

Table of content:

1. Introduction

In our project, we embark on the task of extractive text summarization, leveraging the power of the TextRank algorithm within a Python environment. Our methodology unfolds in distinct stages, beginning with meticulous data preparation. Utilizing NLTK for tokenization, laying the groundwork for subsequent analysis. Moving forward, we delve into model building and training, where the crux of our endeavor resides. Here, we employ advanced techniques to distill the essence of cleaned-up text, crafting concise summaries while exploring the potential of TextRank and graph-based representations. Finally, our evaluation stage presents a pivotal juncture, wherein we gauge the efficacy of our approach. Central to this assessment is the comparison between human-authored summaries and those generated through our model, scrutinized through the lens of the BLEU evaluation metric.

2. Data Preparation

In this section, we detail the steps involved in preparing the data for the text summarization task. The data preparation process encompasses the following steps:

A) Data Collection

Data for this project was generated using an AI-based text generation system, generated autonomously by the AI model.

B) Data Cleaning

In this section, we describe the steps involved in cleaning the raw data obtained from the sources mentioned in the Data Collection section. The data cleaning process includes the following steps:

1. Tokenization:

We used the ‘sent_tokenize()’ function from the NLTK library to tokenize the text into individual sentences.

2. Preprocessing:

- StopWords Removal: We imported a list of stopwords using NLTK's 'stopwords' module and removed them from the sentences. Stopwords often do not carry significant meaning int text summarization tasks.
- Punctuation Removal: We imported 'string' module to access a set of punctuation characters and removed them from the sentences. Punctuation removal helps in focusing on the essential words in the text.
- Stemming: We imported 'WordNetLemmatizer' class from NLTK's 'stem' module and applied stemming to reduce words to their root or base form. Stemming helps in reducing variations of words and consolidating them into single representation, which can improve the task.
- Corpus Creation: We created a corpus by applying the preprocessing steps to each sentence in the tokenized text. A corpus is a collection of preprocessed text data that serves as input for extractive text summarization.

Overall, the data cleaning process involved preparing the text data by removing irrelevant information.

3. Model Building and Training

In this section, we outline the steps involved in constructing and training the text summarization model using the TextRank algorithm and evaluating its performance using BLEU (Bilingual Evaluation Understudy) metrics. We perform the following steps for model building:

1. **Calculate TF-IDF Vectors**: We use the `TfidfVectorizer` from scikit-learn tocalculate the TF-IDF (Term Frequency-Inverse Document Frequency) vectors for each sentence in the corpus. TF-IDF is a numerical statistic used to reflect the importance of a word in a document relative to a collection of documents.

2. **Calculate Cosine Similarity**: We calculate the cosine similarity between each pair of sentences based on their TF-IDF vectors. Cosine similarity measures the cosine of the angle between two vectors and provides a measure of similaritybetween documents.

3. **Build Graph**: We create a graph using the NetworkX library. Each node inthe graph represents a sentence, and the edges between nodes represent the similarity between sentences.

4. **Add Nodes**: We add nodes to the graph, with each node representing asentence from the corpus.

5. **Add Edges**: We add edges to the graph, where the weight of each edgerepresents the cosine similarity between the corresponding pair of sentences.

6. **Calculate PageRank Scores**: We calculate the PageRank scores for each node in the graph. PageRank is a link analysis algorithm used to rank web pagesbased on their importance.

7. **Sort Nodes by PageRank Score**: We sort the nodes in descending order oftheir PageRank scores to identify the most important sentences in the text.

8. **Select Top Sentences**: We select the top-ranked sentences based on their PageRank scores. In this example, we select the top 10 sentences as the summary.

9. **Print Top Sentences**: We print the top-ranked sentences into a separatedocument.

This process results in the selection of the most important sentences from the text, which can be considered as a summary of the document. These top-ranked sentences capture the key information and main ideas present in the text.

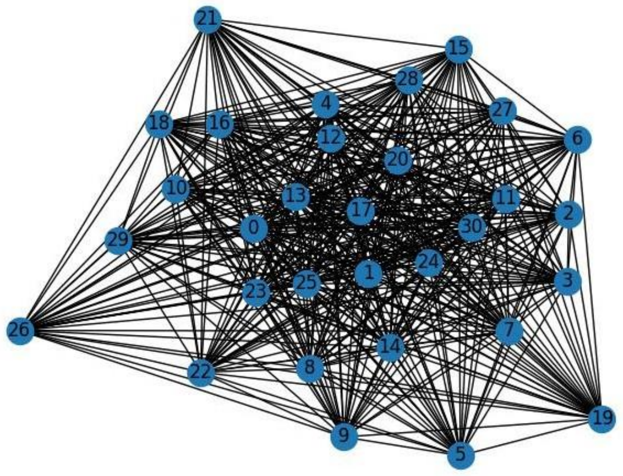
<u>Graph Representations</u>

1. **Cosine Similarity Calculation**: Begin by describing how cosine similarity is

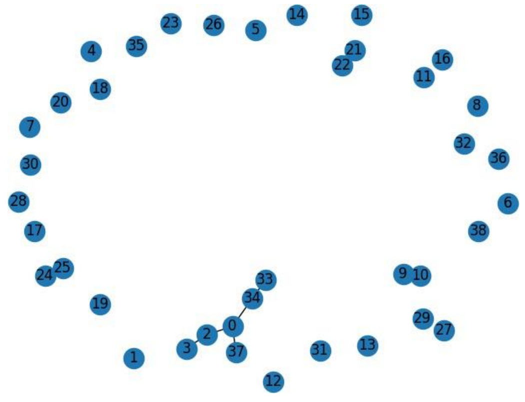
calculated based on TF-IDF (Term Frequency-Inverse Document Frequency) vectors. This involves representing each sentence in the corpus as a vector in a high-dimensional space, where each dimension corresponds to a unique term in the vocabulary. Cosine similarity is then computed between pairs of these vectors to measure the similarity between sentences.

2. **Building Weighted Graphs**: Explain that the cosine similarity scores are used to construct a weighted graph, where each sentence corresponds to a node and the similarity score between pairs of sentences determines the weight of the edge connecting them. Higher similarity scores result in stronger connections (higher weights) between nodes.

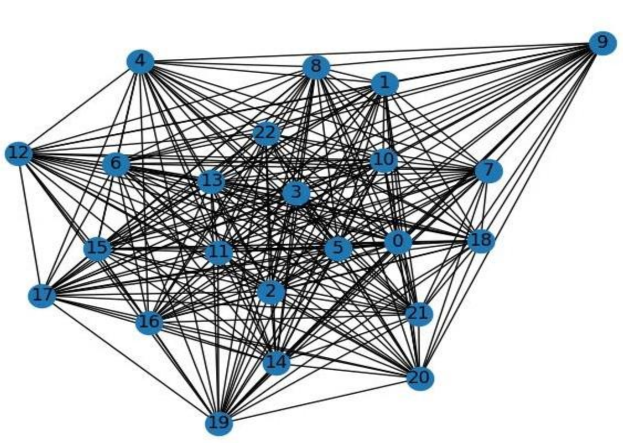
Below are two of the examples of the output graphs we obtained from our model. The thresholding technique was used to remove weak edges from the graphs.

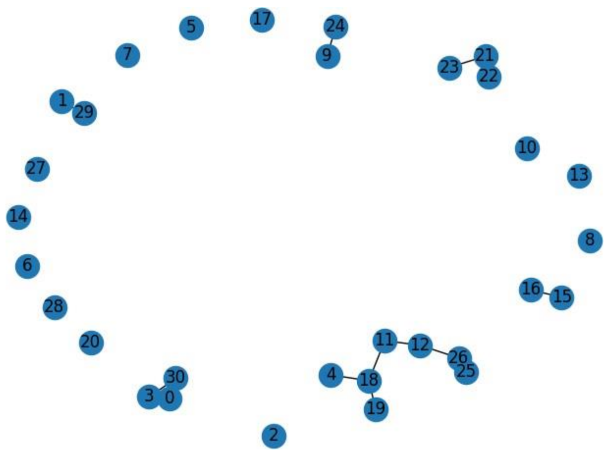


Recycling graph Before thresholding



Recycling graph After thresholding


MentalHealth graph Before thresholding


MentalHealth graph After thresholding

1. **Utilizing Weighted Graphs**: Weighted graphs capture not only the presence of connections between sentences but also the strength of these connections, allowing for more nuanced analysis of sentence relationships.
2. **Implications for Summarization**: By applying graph algorithms such as PageRank, sentences with higher importance scores can be identified, forming the basis for generating extractive summaries.

4. Results

The BLEU score ranges from 0-1, the closer the score is to 1, the higher the similarities between the generated summary and the reference summary. The results below were obtained from comparing human-authored summaries (used as reference summaries) to the summaries generated by our model.

Article	**BLEU Score**
Recycling.txt	0.03815938537306097**5**
Nature.txt	0.065441643871883
Mental health.txt	0.06398031548033703
Artificial Intelligence.txt	0.07327975096283819
Cars.txt	0.171259448396537

The Bleu scores are low, (below 0.5) indicating that the similarities between the generated summary and reference summary are low. The textRank algorithm doesn’t generate accurate summaries. As an alternative, other algorithms can be used.

5. Conclusion

In this study, we embarked on a comprehensive journey to explore the efficacy of textRank and cosine similarity in the domain of text summarization. Through meticulous data preparation, which involved tokenization, preprocessing, and the application of textRank algorithms, we transformed raw text data into a structured format amenable to further analysis.

Utilizing cosine similarity, we constructed weighted graphs to represent semantic relationships between sentences. The visualization of these graphs provided valuable insights into the interconnectedness of sentences within the document. Subsequent thresholding enabled us to refine the graph, filtering out weaker connections and highlighting the most salient relationships.

However, despite our meticulous efforts, the calculated BLEU scores fell below our anticipated thresholds. This unexpected outcome suggests potential limitations in our approach or the need for further refinement in our methodology. Factors such as the granularity of sentence representation, the choice of similarity metrics, and the thresholding criteria may have influenced the summarization quality.

References

J. N. Madhuri and R. Ganesh Kumar, "Extractive Text Summarization Using Sentence Ranking", *2019 International Conference on Data Science and Communication (IconDSC)*.

F. Yamout and R. Lakkis, "Improved TFIDF weighting techniques in document Retrieval", *2018 Thirteenth International Conference on Digital Information Management (ICDIM)*