Summarization of Articles

Advances in AI

Prof. Dr. Dennis Müller

Project Documentation

Kenneth Nelson Djoumessi Takoudjou

897844

Table of Content

Introduction to the Project……………………………………………………………………………………………………………..3

My Motivations…………………………………………………………………………………………………………………………………….4

The Different Techniques ………………………………………………………………………………………………………………..5

My Code and Explanations………………………………………………………………………………………………………………7

Next steps………………………………………………………………………………………………………………………………………………11

Reference……………………………………………………………………………………………………………………………………………….12

**Introduction to the Project**

The project is to be able with the use of programming techniques to summarize a written bulky text into a small compressed readable text that contains all the main information as in the original text.

In today’s information-rich age, the ability to distill vast volumes of text into concise, yet informative summaries is paramount. This challenge forms the crux of my project: harnessing the power of programming techniques and machine learning models to summarize bulky written text effectively. With the growth of data across various domains, the need for efficient summarization methods has never been more pressing.

By employing cutting-edge algorithms and methodologies, I aspire to overcome the challenges inherent in summarization, such as preserving semantic meaning, maintaining coherence, and avoiding information loss. This project holds significant implications across various fields, from academia and journalism to business and research, where the ability to efficiently process and extract insights from large volumes of text can drive productivity and innovation.

**My Motivations**

I have an exceptional motivation to realize this project. The main reason is, that I directly thought of how it could help a lot of people and me especially, thinking of reading many articles or pages to prepare my self for an Exam. This had me thinking at the importance of such a program in our day-to-day life.

I endeavor to bridge this gap by developing an automated summarization system capable of condensing lengthy documents into compressed, readable texts while retaining essential information. Leveraging advancement in natural language processing and machine learning, I seek to create a tool that not only captures the key concepts and insights of the original text but also presents them in a coherent and digestible format.

I was intrigued by the new methods and techniques such as sentence transformers, LexRank, the different types of approaches such as semantic search, and the different others. Through this venture, I envision empowering users with a valuable tool that streamlines the consumption of information, enabling them to make informed decisions, extract insights, and stay abreast of developments within their respective domains. Mainly for Students having problems reading large volumes of text to prepare for their exam sessions, having more than 5000 pages of text in total to read within a small range of time.

I as a student in data science and AI was very pleased and wanted to bring this project to the world and my friends, offering me the possibility to learn along the way and create something that everyone can enjoy.

**The Different Techniques**

There are various techniques and methods today due to the evolution of the world, that ease the way to realize such a project.

To begin with, we have to explain what Natural Language Processing is: Natural Language Processing stands as a pivotal technology in the realm of AI, bringing the gap between human communication and computers. NLP is a branch of AI that focuses on the interaction between computers and humans through natural language.

To analyze and understand human language, NLP employs a variety of techniques and methods. Here are some techniques:

1. **Tokenization**: This is the process of breaking text into words, phrases, symbols, or other meaningful elements, known as tokens.
2. **Parsing**: It involves analyzing the grammatical structure of a sentence to extract meaning.
3. **Lemmatization**: This technique reduces words to their base root form, allowing for the grouping of different forms of the same word
4. **Sentiment analysis**: This method is used to gain an understanding of the sentiment or emotion conveyed in a piece of text.

Coming to the text summarization, there are two primary methods, namely:

1. **Extractive Text Summarization**: With this method, notable information is extracted directly from the original text. Key Sentences are selected based on predetermined parameters, such as importance or relevance. The most common technique is the ***TextRank algorithm***, which treats sentences as vertices in a graph and ranks them based on their global importance within the graph.

Gensim: It is an open-source topic and vector modeling toolkit in Python. It provides a variation of the TextRank algorithm for extractive summarization.

LexRank: It is based on the concept of eigenvector centrality in a graph representation of sentences. Within this algorithm, each sentence recommends sentences similar to it. A graph is created with each node being a sentence, connected to its similar sentences ( the similarity measure is usually Cosine Similarity or TF-IDF). Sentences with maximum recommendations is more likely to get picked for summary.

1. **Abstractive Text Summarization**: It generates new sentences that capture the essence of the original text. It involves rewriting and creating acceptable representations. It’s more akin to AI-generated writing. It Is mainly based on the following steps: firstly, Establishing a context for the text ( This step is to understand the context of the text) and lastly, semantics ( Words based on semantics understanding of the text are either reproduced from the original text or newly generated.).

**My Code and Explanations**

To write my code, I did at first some research on the various techniques that I had given before, and I decided to implement the Extractive Approach. For me, it is the best approach because it uses exactly what the author of the text wants to send as a message and the fact that this approach just takes the most relevant sentences in the text as a summary without adding or deleting the words and phrases. The other method which is more of interpreting the text and generating new phrases was not the most optimal method for me as there is a high possibility of error-making while interpreting.

Meanwhile, with the extractive approach, there are many sub-methods, I opted for the LexRank, NLTK, and TF-IDF (sklearn) methods.

To start, here is the LexRank code I used to compute the degree centrality score of each sentence based on a similarity matrix to have the most relevant summary.

Ein Bild, das Text, Screenshot enthält.

Automatisch generierte Beschreibung

The algorithm creates a graph of sentences, where each sentence is a node, and the edges represent the similarity between sentences. The ***degree\_centrality\_scores*** function calculates the degree centrality score for a similarity matrix. The score of a node is the sum of the weights of the edges incident to that node. Here a Markov matrix is used which is the similarity matrix and the stationary distribution is calculated. This distribution represents the probability of being in each state in the long run. It decides if it creates a discrete Markov Matrix or a continuous based on if there is a threshold

Ein Bild, das Text, Screenshot enthält.

Automatisch generierte Beschreibung

To continue, the ***power\_method*** function calculates the stationary distribution of a Markov matrix using the power method, which is an iterative algorithm for finding the eigenvector corresponding to the largest eigenvalue of a matrix.

The connected-nodes function finds the connected components of a graph represented by a matrix. It is a subgraph where all the nodes are connected. This function returns a list of indices for each connected component.

***Create\_markov-matrix*** function creates a Markov matrix. This matrix is a square matrix where each row sums to 1. If the weights contain negative values it normalizes them using softmax or by row sums

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

The ***Create\_markov\_matrix\_discrete*** function creates a discrete Markov by thresholding the input weight matrix. This discrete Markov matrix is like the continuous Markov matrix but here each element is either 0 or 1.

Lastly, with the LexRank code, the function ***stationary\_distribution*** calculates as said before the stationary distribution.

Based on this LexRank code the decision of how I wanted to write my main code came in place.

Ein Bild, das Text, Screenshot, Schrift, Software enthält.

Automatisch generierte Beschreibung

Here are the different libraries that I needed to write my main code. After importing the libraries, I downloaded the necessary resources from NLTK. Nltk.download(‘punkt’) downloads the Punkt tokenizer models for tokenizing natural language text. In other words, it is splitting text into individual words or sentences. It is a pre-trained model capable of handling various language and is commonly used in NLP tasks. Nltk.download(‘stopwords’) downloads a list of common stopwords for various languages. Examples in English include “the”, “is”, etc.

Ein Bild, das Text, Screenshot, Software enthält.

Automatisch generierte Beschreibung

I decided the Article dataset from Kaggle because of the amount of information there is in there and the length of the texts. They are long texts and provide more accuracy to my code and my embeddings. And there are no missing objects or NaN in the dataset.

The main function takes a list of texts as input from my dataset and returns a list of summaries for each text. I initialize an empty list of ‘*summaries’ to* store the summaries for each text. It tokenizes the text into sentences using the nltk.sent\_tokenize function. We can also say that it splits each bulky text into sentences. Compute the sentence embeddings using the ‘TfidfVectorizer’ from the sklearn library. It converts each sentence into a vector of TF-IDF features. The ‘stop\_words’ parameter is set to the English stop words to remove common words that do not contribute much to the meaning of the sentence just like an auto-corrector. After that, it computes the pair-wise cosine similarities between the sentence embedding. We need the pair-wise computation for our LexRank method in order to have more precise and accurate sentences based on their degree centrality score.

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

This code block implements batch processing of text summarization for the large number of texts stored in the DataFrame. The idea behind this is to, first of all, calculate the number of batches needed to summarize all the texts in the DataFrame. Then starts a loop that processes each batch of texts by selecting the batch of texts to be summarized and selects a slice of the column based on the index, then applies the main function to each batch of texts and appends the resulting summaries to the list.

To conclude the code, I created a new DataFrame called *result\_df* containing Authors, Titles, and summaries respectively.

**Next Steps**

This code is a very future-related thing since the world of Natural Language Processing and machine learning is improving. In the future to optimize and improve my project I can try the Abstractive Approach which consists of methods like OpenAI’s GPT-3 API. Furthermore, the most reliable Sentence transformer models like the ‘all-MiniLM-L6-v2’ model for semantic search and the use of numba and Dask.

**Reference**

[UKPLab/sentence-transformers: Multilingual Sentence & Image Embeddings with BERT (github.com)](https://github.com/UKPLab/sentence-transformers?tab=readme-ov-file)

[(11799) NLP simple text summarization with python programming - YouTube](https://www.youtube.com/watch?v=ZP0RgW1CKiw&ab_channel=ShabbirGovernor)

[crabcamp/lexrank: LexRank algorithm for text summarization (github.com)](https://github.com/crabcamp/lexrank)

[5 Powerful Text Summarization Techniques in Python. (turing.com)](https://www.turing.com/kb/5-powerful-text-summarization-techniques-in-python)

[crabcamp/lexrank: LexRank algorithm for text summarization (github.com)](https://github.com/crabcamp/lexrank)