**TEXT SUMMARIZATION TOOL USING NLP**

**ABSTRACT:** This project focuses on implementing extractive text summarization techniques using natural language processing in python.The aim is to automatically generate concise and informative summaries of article by identifying and extracting the most important sentences. Although the concept of text summarization started decades and decades ago in the 1950s, the field has continued to remain challenged in providing the best and as efficient summaries as possible. Text summarization proceeds into 2 methods which are Explanative Summarization and Abstractive Summarization. The methods of Explanative and Abstractive summarization, have their individual processes to better the textual summarisation methods.Text Rank is something to extractive text summarization and is an unsupervised learning. Text Rank algorithm also is used for undirected graphs ,weighted graphs, keyword extraction, sentence extraction. So, in this project , a model is developed to acquire better text summarization

**INTRODUCTION:** With the rise of digital text in the form of documents and websites, it is crucial to develop tools that can automatically compress large quantities of digitally written text in a way that retains core meaningful content. Automated text summarization, specifically extractive text summarization, is able to meet this need by selecting the most relevant sentences with respect to the original document.

There is a lot of information that is on the internet, and the volume is ever increasing, this is relevant to news too, which means there is too much content for people to be able to recollect and sort out into main points. Hence the emergence of automatic text summarization, as this can allow offer up important points from large amounts of information. Some of the text graph based ranking algorithms of the top actions we can take to create assign points as relevant in the Text Rank algorithm. Writing down the important points, as a note taking action across this many documents can be a very tedious job. So, automatic text summarization can take and pull the most important words and return this back in a way that readers feel informed. So in summary, automatic text summarization is a small contributor to deciding what information is useful and worthy of the user time.

**PROBLEM STATEMENT:** In our busy schedule it is very difficult for us to go through the entire article or document. So we prefer to read summary.

Difficulties Associated With Reading a Long Article:

**Time Consumption:**

Reading a long article takes a lot of time for a reader to read and understand -- and this alone is too much for some readers.

**Information Overload**:

Readers may have a hard time remembering the main points they want to retain because of the sheer volume of text, which may sometimes lead to misinterpretation or misunderstanding of crucial detail.

**Difficulties Identifying Key Points**: When a reader approaches a long article, it can be difficult to determine what points are the most important, or main, points without some type of approach.

**OBJECTIVE:**

* To develop an extractive summarization tool using frequency based and graph based (text rank) methods.
* To implement preprocessing steps including tokenization,stopword removal,and sentence segmentation.
* To evaluate and compare summaries across different articles in the dataset.

**SCOPE:**

* Summarization is limited to English news articles.
* Only extractive summarization methods are used.
* The model is designed for offline, batch summarization—not real-time or streaming applications.
* TextRank is introduced but not fully implemented in this version. Future scope includes extending it to weighted graph-based ranking.

**DATASET DESCRIPTION:** The data set being used is a CSV set called IK\_Trump.csv that has many news articles in 'ArticleText'. Each news article is handled independently and referenced via an index for summaries. The data encoding 'latin-1' is loaded using Pandas. Latin-1 specifies how characters (like letters, numbers, and symbols) are represented as bytes.

It has 256 characters: 128 are exactly the same as ASCII (these are the basic English letters and symbols), and the next 128 represent characters used in Western European languages (like é, ñ, ü etc.).When reading a text file(.csv), python needs to know what encoding was used to store character. If the file contains special european characters,encoding=’latin-1’ tell python how to interpret those characters correctly.

**DATA PREPROCESSING:** The preprocessing pipelines involve:

* Tokenization
* Stopword removal
* Punctuation removal
* Word frequency calculation
* Sentence scoring

**Tokenization:** Breaking words into smaller units such as words(word tokenization) and sentence(sentence tokenization). Machine learning and nlp models need to work with units of language not raw text.

* SpaCy handles tokenization text into words and sentences using its language model.
* NLTK is used to split paragraphs into sentences using sent\_tokenize()

**Stopword Removal:** Removing common less meaningful words like “the”, “is”, “in” etc. stopwords appear frequently but dont add significant meaning to a text; removing them reduces noise in the analysis.

* SpaCy provides a built in list of stopwords with STOP\_WORDS.
* NLTK has its own stopword list under nltk.corpus.stopwords.

**Punctuation Removal:** It is used to eliminate characters like **. , ! ? ; :** from the text. Punctuation does not carry meaningful weight for summarization and can interfere with frequency analysis

* Uses python string. Punctuation set along with newline characters(\n)
* Each token is checked and filtered out if its punctuation.

**Word Frequency Calculation:**It is used to count how often each word appears in the article . words that appears more often are likely to be important for understanding the main idea.

* Loop through each token and update a frequency dictionary.
* Then normalize these frequency by dividing each count by the maximum frequency value(to bring into [[0,1] range).

**Sentence Scoring:** Assigning a score to each sentence based on the frequencies of the word it contains. Higher score indicates that the sentence contains more important or frequently occurring words.

* Loop through each sentence.
* Add up the normalized frequencies of the words in that sentence.
* Store the total score in a dictionary with the sentence as the key.

**TEXT SUMMARIZATION IN NLP:** Text summarization aims to shorten one or multiple texts into shorter summaries, so that information extraction is improved.

Automatic text summarization (or document summarization) is a natural language processing (NLP) technique that reduces information from one or multiple input text documents into a novel output text. How much of the input text appears in the output is debatable: some definitions state that 10% should appear in the summary, and others state that it should be 50%.1 Text summarization algorithms generally use deep learning architectures (in particular, transformers) to parse documents and generate text summaries.

Extractive summarization includes sentences taken literally from the documents of original text. A salient difference among extractive algorithms is in how they score importance of sentences that reduces topical redundancy. Sentence scoring differences dictate the extraction of certain sentences and their retention.

Abstractive summarization is composed of newly generated summarization using entirely new original sentences not found in the accordance of rules from the original text document. Such generation requires neural networks and large language models (LLMs) to produce semantically meaningful text sequences.

**METHODOLOGY:**

**TEXT -RANK ALGORITHM:**Text rank algorithm is a graph-based ranking system for text processing which can be used to identify the most relevant sentences in text and to identify keywords. Text rank algorithm is similar to page rank algorithm. Page rank algorithm is a graph-based ranking system which has been developed to rank web pages in web search engines and in web usage mining. In text rank algorithm, sentences are used in place of web pages. 1. Identify content units that most representative of the task at hand and add them as vertices in the graph. 2. Identify the relationships that join the content units, and in the graph use those relationships to draw edges between the vertices. Edges can be un-weighted or weighted and undirected or directed. 3. And then iterate the graph-based ranking algorithm until convergence. 4. Rank the vertices based on their final score. Use the attributes in each vertex for ranking and selection decisions. 5. In the end, the top-ranked sentences will form a summary.

TextRank is an unsupervised graph-based algorithm patterned after Google's PageRank that uses sentences as nodes in a graph and connects them based on their semantic similarity. Sentences that are more central (i.e. connected to other important sentences) have a higher rank and are included in the summary.

How It Works

* Break the text into sentences.
* Calculate similarity between each pair of sentences (for example: cosine similarity or by using SpaCy's document vectors similarity).
* Construct the graph: the nodes are the sentences and the edges are the similarity weights.
* Run the PageRank algorithm and score the sentences.
* Select the sentences with the highest score to create the summary.

**FREQUENCY BASED APPROACH:** The model selects those sentences from the original text that have the highest frequency of words. The assumption is that more frequently occurring words in a document are the most relevant ones and that the sentences containing those words are informative.

How It Works

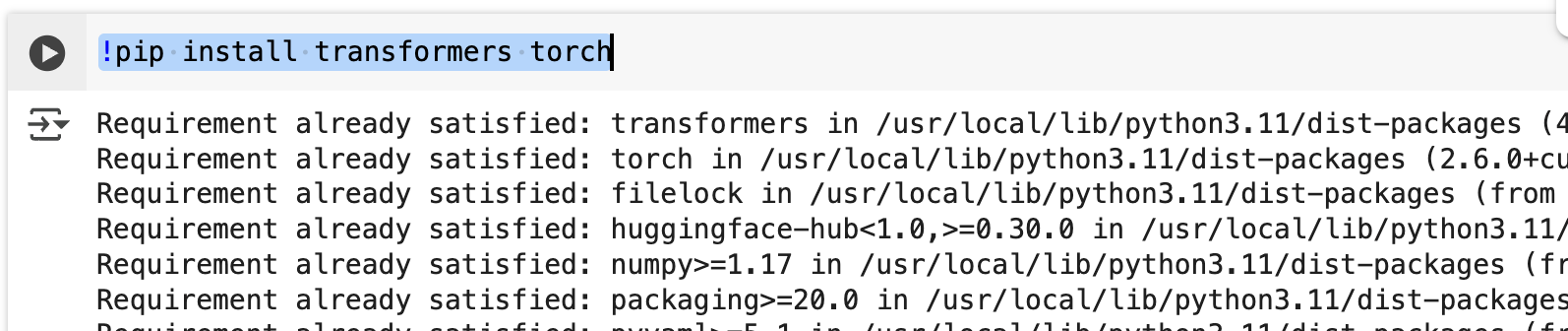
* The text is first broken down into words and sentences.
* Stopwords and punctuation are removed.
* Create Dictionary for Words: Count how many times each word occurs.
* Normalize frequency of each word by dividing it by the highest frequency word.
* Score sentence: By summing the frequency of the words in the sentences.
* Select top N scoring sentences for the summary.

**BART:** BART (Bidirectional and Auto-Regressive Transformers) is a powerful transformer-based model developed by Facebook AI that also processes sequence-to-sequence tasks such as abstractive text summarization. BART takes advantage of both BERT (contextual understanding) and GPT (text generation).

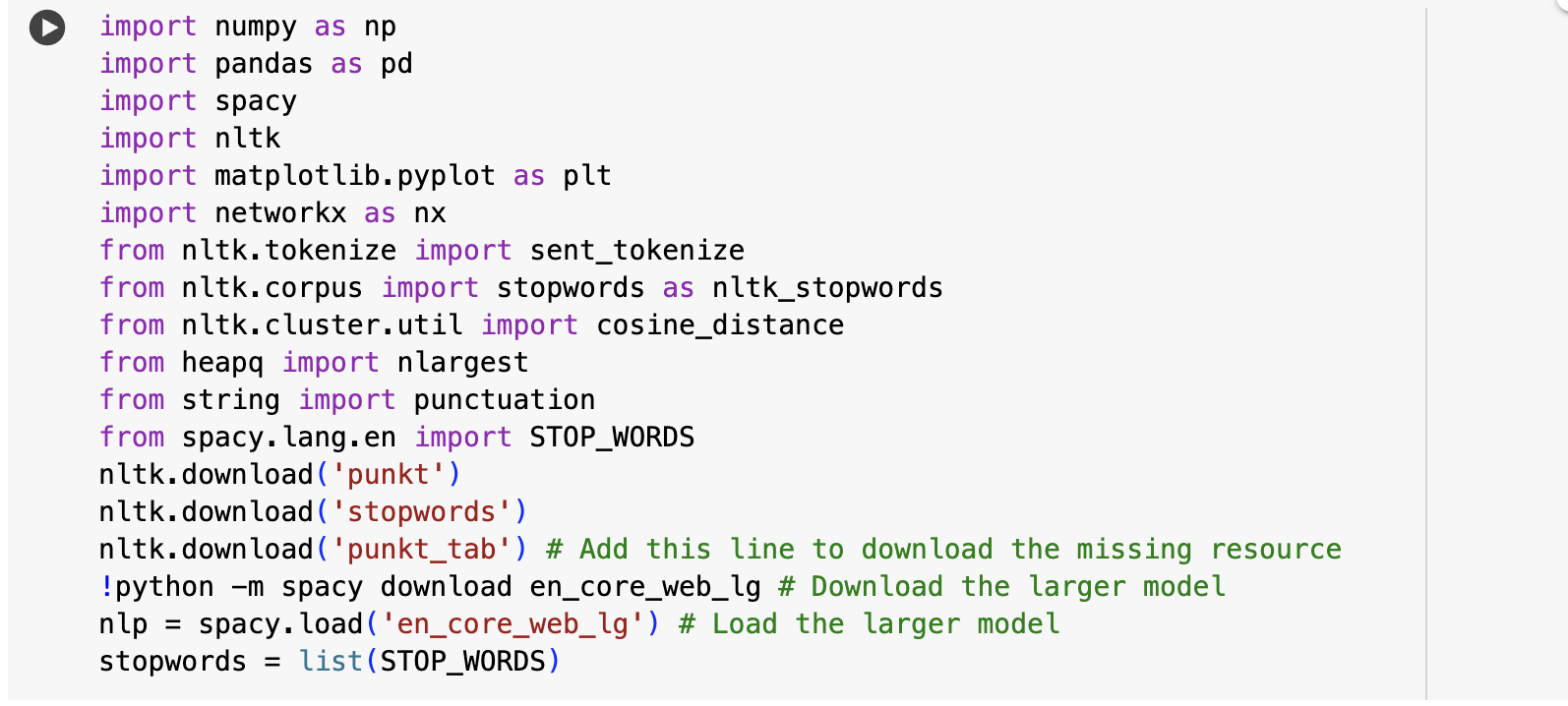
It produces fluent, grammatically correct, and context-aware abstractive summaries.

* Supports fine-tuning on summarization datasets like CNN/DailyMail,triumph .
* Pretrained models like facebook/bart-large-cnn are available and widely used for summarizing long documents.

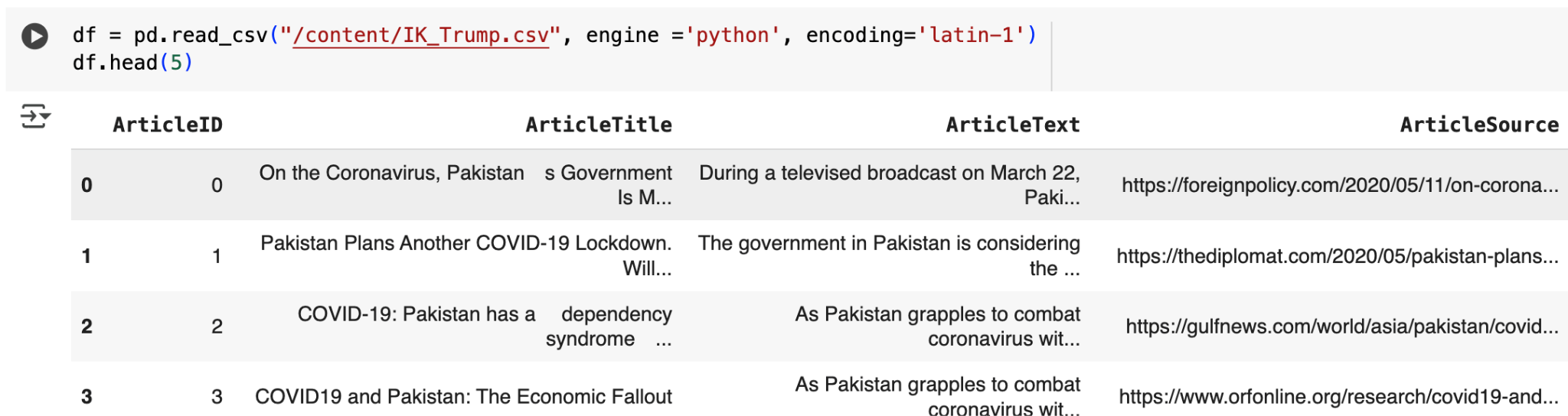
**IMPLEMENTATION STEPS:**

****

* Install the necessary libraries transformer is a hugging face library it provides a pre trained model like BART pipeline function comes under this library.



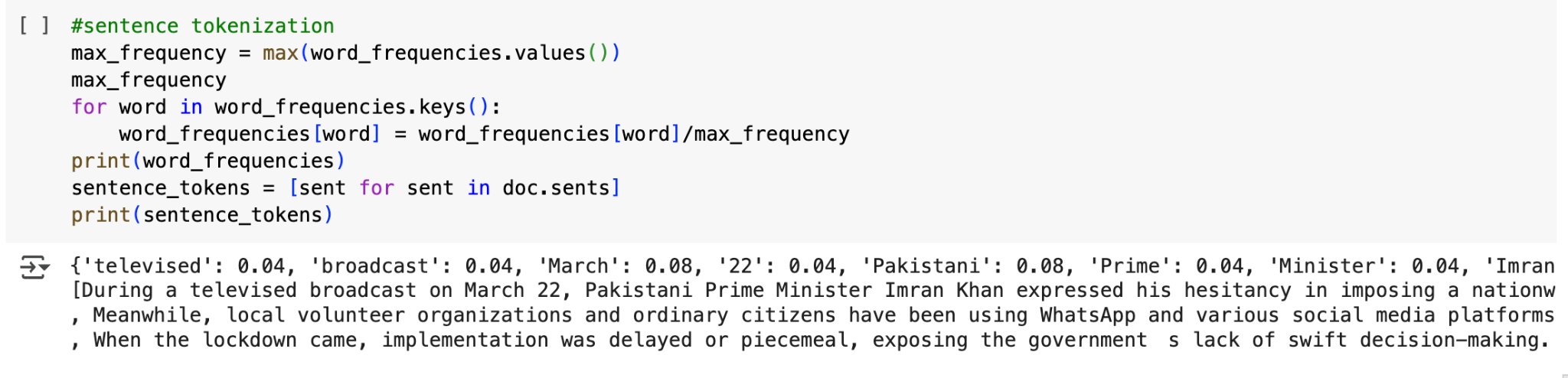
* This code imports the necessary libraries and resources required for NLP-oriented text summarization using:
* NLTK and SpaCy for text preprocessing functions (tokenization, stopword removal, embeddings).
* other libraries including numpy, pandas, networkx, matplotlib for data manipulation, graph algorithms (e.g., TextRank) and for visualizing.
* It downloads the NLTK resources (punkt, stopwords, punkt\_tab).
* It also installs and loads the SpaCy larger English model (en\_core\_web\_lg) that comes with richer word vectors useful for sentence/similarity based summarization.

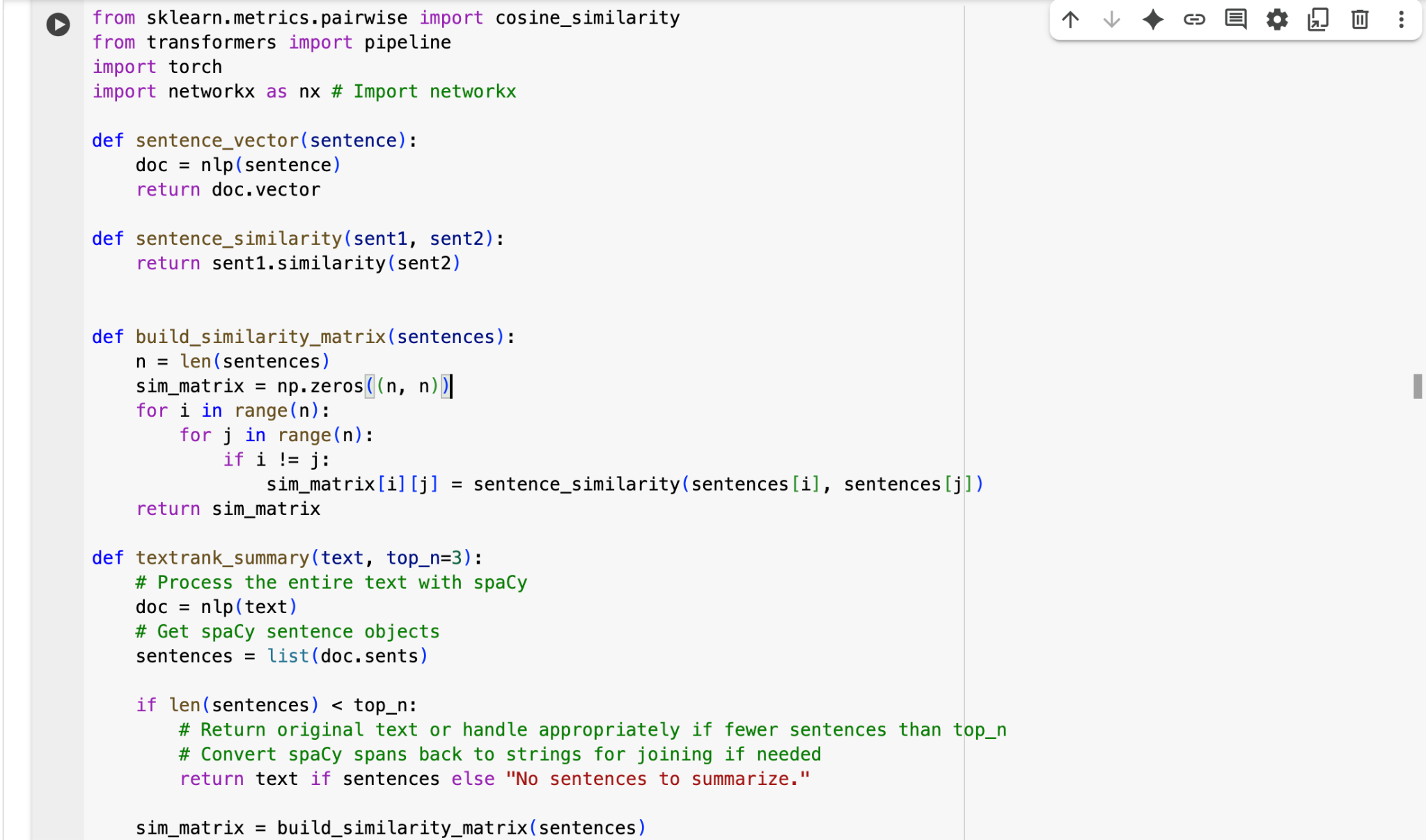


Import the necessary dataset for text summarization which consists of columns like article id, article title, article text, article source.



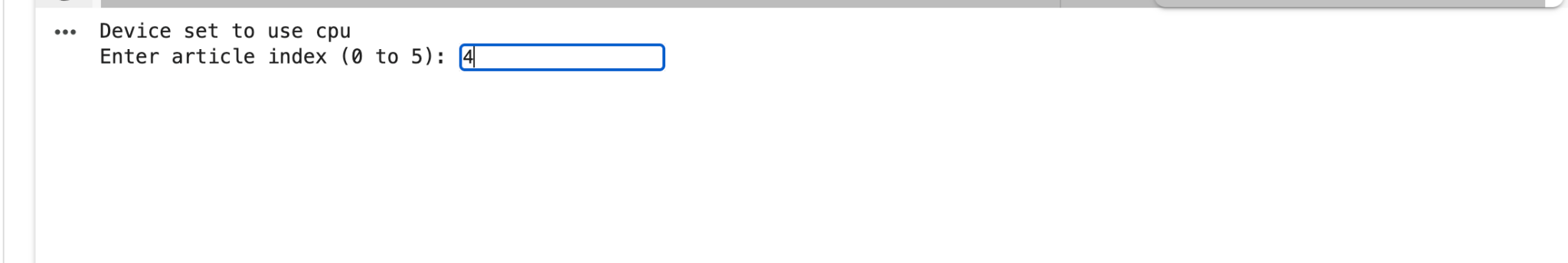
The following function perform word tokenization and frequency calculation using SpaCy on the first article's text from the dataset. It filters out stopwords and punctuation, then builds a dictionary of word counts (word\_frequencies).

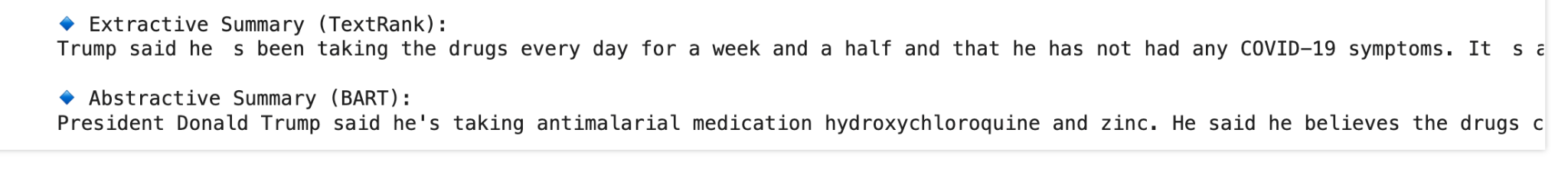


In this code, word frequencies are normalized by dividing each by the maximum frequency, and then SpaCy is used to tokenize the article into sentences. Finally, it prints the normalized word frequencies and the list of sentence tokens extracted.

These two code blocks implement both extractive (TextRank) and abstractive (BART) summarization methods. The first builds a sentence similarity matrix using SpaCy vectors for ranking, while the second uses the BART model via Hugging Face’s pipeline for generating concise summaries from selected articles.

**OUTPUT:**

****

****

* The output contains two summaries from two different approaches.
* The Extractive Summary (TextRank) literally picks sentences from the article, using the original words—and there are still minor artifacts like spacing issues.
* The Abstractive Summary (BART) rewrites everything with natural language generation producing a more fluid and concise summary.
* BART's output contains specific terms like "hydroxychloroquine and zinc," this exhibit greater contextual understanding of this term in the summary.
* Ultimately, the abstractive summary is better to consume and flows better while the extractive summary remains a little closer to the text.

**CONCLUSION:**The paper shows that we used sophisticated techniques to apply on the document for text summarization using an extractive summarization method called TextRank algorithm. Initially, we loaded necessary libraries and related function in python, and then implementing the code to summarizes the text. Subsequently, a model (as shown) was proposed, with a few enhancements for clarification, which suggests the outline given by the document. The techniques demonstrated in this paper can provide us in the aggregate a better result in summarizing the text, At a glance the overall meaning of the document can be established quickly and easily. This project efficiently demonstrates a basic but effective implementation of extractive text summarization using Python. Frequency-based approaches offer a fast and straightforward way to condense text. While the TextRank logic is included, it does not implement the full graph-based logic in this version of the text, and therefore can be improved. Overall, the summaries generated were informative and utilizeable as quick reads to obtain information.

**FUTURE WORK:**

* Full implementation of TextRank using NetworkX for graph generation and PageRank computation.
* Incorporate cosine similarity between sentence vectors to strengthen the TextRank scoring mechanism.
* Add abstractive summarization using Transformer-based models like BERT or T5.
* Evaluate summaries using ROUGE or BLEU scores to assess performance.
* Build a GUI or web app for broader usability and live summarization.