

INTERDISCIPLINARY PROJECT REPORT
at
Sathyabama Institute of Science and Technology
(Deemed to be University)

Submitted in partial fulfillment of the requirements for the award
of Bachelor of Engineering Degree in Computer Science and
Engineering

By

NAME: YASHWIN P PRAKASH
(Reg.No. 40731132)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU

APRIL 2023



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI- 600119

www.sathyabamauniversity.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of
Yashwin P Prakash (40731132) who carried out the project entitled
"Youtube Summarization Using TFIDF Vectorization and BART " under my
supervision from FEB 2023 to APRIL 2023.

Internal Guide

Dr. Veena K M.E., Ph.D.

Head of the Department

Dr.S.Vigneshwari M.E., Ph.D.

Submitted for Viva voce Examination held on 17-04-2023

Internal Examiner

External Examiner

DECLARATION

I **Yashwin P Prakash** hereby declare that the Project Report entitled '**YouTube Summarization Using TFIDF Vectorization and BART**' Done by me under the guidance of **Dr. Veena K** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE: 17-04-2023

PLACE: Chennai



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D., Dean**, School of Computing, **Dr.S.Vigneshwari M.E., Ph.D.**, Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. Veena K** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

COURSE CERTIFICATE

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	1
	LIST OF FIGURES	2
	LIST OF ABBREVIATIONS	3
1	INTRODUCTION	11
	1.1 PROJECT AIM AND OBJECTIVES	11
	1.2 BACKGROUND OF PROJECT	12
	1.3 OPERATION ENVIRONMENT	12
	1.4 OVERVIEW OF PROJECT	12
2	AIM AND SCOPE OF THE PRESENT INVESTIGATION	13
	2.1 AIM	13
	2.2 SCOPE	13
	2.3 SOFTWARE TOOLS USED	13
	2.3.1 PROGRAMMING LANGUAGE	
	2.3.2 LIBRARIES USED	
3	EXPERIMENTAL OR MATERIALS AND METHODS	14
	3.1 THEORY	14
	3.1.1 EXTRACTIVE SUMMARIZATION	14
	3.1.2 ABSTRACTIVE SUMMARIZATION	14
	3.2 TFIDF VECTORIZATION	14
	3.2.1 TERM FREQUENCY	15
	3.2.2 INVERSE DOCUMENT FREQUENCY	16
	3.2.3 ADVANTAGES OF TFIDF VECTORIZATION	16
	3.2.4 DISADVANTAGES OF TFIDF VECTORIZATION	16
	3.3 BART	17
	3.3.1 TRAINING OF BART MODEL	18
	3.3.2 PARAMETERS OF BART	19

		21
4	SYSTEM ARCHITECTURE	
5	CONCLUSION AND SUMMARY	23
	5.1 CONCLUSION	23
	5.2 SUMMARY	23
6	SOURCE CODE AND SCREENSHOTS	24
7	REFERENCES	38

ABSTRACT

In this article, we are discussing a mini-NLP project, a YouTube Summarizer which will summarize the content(subtitle) of the YouTube video. For many videos, the main content of the videos is only 50-60% of the total length, so our YouTube summarizer will summarize the content of the video by keeping all the important points and making it short and easily understandable. This will be useful in getting the summary of several lecture videos easily.

Summarization is the technique of making short, understandable notes for a given large text document without excluding the important contents of the passage. There are 2 types of summarizations in NLP, extractive summarization and abstractive summarization.

In **extractive summarization**, the system will extract the important paragraphs and contents from the given passage and combine these extracted paragraphs to create the summarized text.

In **abstractive summarization**, the system will create a summary based on the given passage with its own words. This is more complex than extractive summarization.

YouTube is the world's largest video sharing platform, with over 2 billion active users. With such a vast amount of content available on YouTube, it can be challenging to find relevant videos that align with specific interests. This is where summarization techniques come in handy.

In this presentation, we will explore how TFIDF vectorization and BART can be used for YouTube summarization. We will discuss the benefits of these techniques and provide examples of how they can be applied.

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
3.1	TD-IDF architecture	15
3.2	BART training model	16
3.3	BART architecture	18
4.1	System architecture	19

LIST OF TABLES

TABLE NO	TABLE NAME	PAGE NO
3.1	BART model parameters	17
3.2	BART Vs Roberta	17

LIST OF ABBREVIATIONS

NLP	Natural Language Processing
GPT	Generative Pre-trained Transformer
TFIDF	Term Frequency Inverse Document Frequency
BART	Bidirectional Auto Regressive Transformers
NLTK	Natural Language Toolkit

CHAPTER 1

1. INTRODUCTION

YouTube is one of the most popular platforms for sharing and consuming educational and informative videos. With millions of videos available on the platform, it can be challenging for users to extract and summarize key information from them quickly and efficiently. Manual summarization of videos can be time-consuming and often leads to information loss. To address this challenge, we propose a YouTube summarizer using TF-IDF vectorization and BART that can automatically extract and summarize information from YouTube videos. The proposed summarizer will use the TF-IDF technique to extract important words and phrases from video transcripts. TF-IDF assigns weights to each word based on its frequency in the transcript and its importance in the overall corpus of transcripts. The technique is widely used in information retrieval and text mining applications and has been shown to be effective in extracting important information from text documents.

Once important words and phrases have been extracted from the video transcript, they will be fed into a BART model, which will generate a summary of the most important information. BART is a pre-trained sequence-to-sequence transformer model that has been shown to produce high-quality summaries for a variety of applications.

The proposed YouTube summarizer has several potential benefits. First, it can save users time and effort by automatically summarizing the most important information from YouTube videos. Second, it can help users retain more information from videos by providing them with concise summaries. Third, it can improve accessibility by providing summaries for users who may have difficulty understanding spoken language or following video content.

The project's scope includes scraping transcripts from YouTube videos using YouTube's API, pre-processing the transcript data, using TF-IDF vectorization to extract key information from the transcript data, training and fine-tuning a BART model on the extracted information to generate summaries, and evaluating the performance of the summarizer using metrics such as ROUGE and human evaluation. In conclusion, the proposed YouTube summarizer has the potential to be a valuable tool for anyone looking to extract and summarize key information from YouTube videos quickly and efficiently. By using TF-IDF vectorization and BART, the summarizer can generate high-quality summaries that capture the most important information from videos, improving accessibility and user experience on the platform.

1.1 PROJECT AIM AND OBJECTIVES

The project aims and objectives that will be achieved after completion of this project are discussed in this subchapter. The aims and objectives are as follows:

- Make all the system computerize.
- Reduce time consumption.
- Reduce error scope.
- All system managements are automated.
- Centralized database management
- Easy operations for operator of the system

1.2 BACKGROUND OF PROJECT

With the increasing popularity of YouTube as a platform for education and information sharing, there is a growing need for tools that can help users quickly extract and summarize key information from YouTube videos. However, manually watching and summarizing videos can be time-consuming and tedious, especially for longer videos. To address this need, natural language processing (NLP) techniques such as TF-IDF vectorization and summarization models like BART can be used to automatically extract and summarize information from video transcripts. TF-IDF is a widely used technique for extracting important words and phrases from text, while BART is a powerful pre-trained transformer model that can generate high-quality summaries.

The proposed YouTube summarizer using TF-IDF vectorization and BART has the potential to benefit a wide range of users, including students, researchers, and professionals who need to quickly extract and summarize information from YouTube videos. The summarizer can save users time and effort, while also improving their ability to comprehend and retain important information from videos.

1.3 OPERATION ENVIROMENT

- **MEMORY AND DISK SPACE REQUIRED:** 1GB RAM + 1GB of disk + . 5 CPU core.
- **SERVER OVERHEAD:** 2-4GB or 10% system overhead (whatever is larger), . 5 CPU cores, 10GB disk space.
- **PORT REQUIREMENTS:** Port 8000 plus 5 unique, random ports per notebook.

CHAPTER 2

AIM AND SCOPE OF THE PRESENT INVESTIGATION

2.1 AIM

The aim of this investigation is to develop a YouTube summarizer using TF-IDF vectorization and BART to automatically extract and summarize key information from YouTube videos.

2.2 SCOPE

1. Scraping transcripts from YouTube videos using YouTube's API.
2. Pre-processing the transcript data to remove any irrelevant information (e.g. timestamps, speaker names, etc.).
3. Using TF-IDF vectorization to extract key information from the transcript data.
4. Training and fine-tuning a BART model on the extracted information to generate summaries.
5. Evaluating the performance of the summarizer using metrics such as ROUGE and human evaluation.

2.3 SOFTWARE TOOLS USED

2.3.1 PROGRAMMING LANGUAGE:

Python (Jupyter notebook)

2.3.2 LIBRARIES USED:

YouTubeTranscriptApi
Nltk
Re
nltk.corpus
stopwords
youtube-transcript-api sentencepiece
TfidfVectorizer
pandas
transformers
AutoTokenizer
AutoModelForSeq2SeqLM
ipywidgets
torchvision

CHAPTER 3

EXPERIMENTAL OR MATERIALS AND METHODS; ALGORITHMS USED

3.1 THEORY

NLP or Natural Language Processing is one of the fastest-growing tech fields right now. From message spam filter to medical diagnosis with a chatbot, NLP is everywhere. Some of the hot use cases of NLP right now are text summarization, chatbot, machine translation, text generation, etc.

YouTube is the world's largest video sharing platform, with millions of videos being uploaded every day. With such a huge amount of content, it can be difficult for users to find the specific information they are looking for. This is where a YouTube summariser comes in. A summariser can take a long video and condense it into a shorter, more digestible format.

YouTube Summarizer which will summarize the content(subtitle) of the YouTube video. For many videos, the main content of the videos is only 50-60% of the total length, so our YouTube summarizer will summarize the content of the video by keeping all the important points and making it short and easily understandable. This will be useful in getting the summary of several lecture videos easily.

Summarization is the technique of making short, understandable notes for a given large text document without excluding the important contents of the passage. There are 2 types of summarizations in NLP, extractive summarization and abstractive summarization.

In **extractive summarization**, the system will extract the important paragraphs and contents from the given passage and combine these extracted paragraphs to create the summarized text.

In **abstractive summarization**, the system will create a summary based on the given passage with its own words. This is more complex than extractive summarization.

For our YouTube summarizer, we are using extractive summarization. For extractive summarization, we are using different summarization techniques like TF-IDF vectorizer and BART. We will first discuss what TF-IDF vectorizer and BART are and how they work. Then, we will go through the steps involved in building a YouTube summariser using these tools.

3.1.1 EXTRACTIVE SUMMARIZATION

It simply takes out the important sentences or phrases from the original text and joins them to form a summary. a ranking algorithm is used, which assigns scores to each of the sentences in the text based on their relevance to the overall meaning of the document. The most relevant sentences are then chosen to be included in the summary.

There are various ways through which the ranking of sentences can be performed.

- TF-IDF (term frequency-inverse document frequency)
- Graph-based methods such as Text Rank
- Machine learning-based methods such as Support Vector Machine (SVM) and Random Forest.

The main motive of the extractive method is to maintain the original meaning of the text. Also, this method works well when the input text/content is already in a well-structured manner, both physically and logically, just like the content in newspapers.

3.1.2 ABSTRACTIVE SUMMARIZATION

Abstractive summarization is a technique in natural language processing that generates a summary of a text by producing new sentences that may not be present in the original text. Unlike extractive summarization, which selects and rearranges existing sentences from the text to create a summary, abstractive summarization can generate novel sentences that capture the main ideas of the original text in a concise and coherent manner. Abstractive summarization is a challenging task that requires deep understanding of natural language and context, and has been an active area of research in recent years.

3.2 TFIDF VECTORIZATION

TFIDF stands for Term Frequency-Inverse Document Frequency. It is a technique used to determine the importance of a word in a document by comparing its frequency in the document to its frequency in the entire corpus of documents. The more frequently a word appears in a document and the less frequently it appears in the corpus, the higher its TFIDF score.

TFIDF vectorization involves converting a document into a vector representation based on its TFIDF scores. This allows us to compare the similarity between documents using cosine similarity, which measures the angle between two vectors. Documents with similar content will have a smaller angle between their vectors, indicating a higher degree of similarity.

To understand TF-IDF, firstly we will understand the two terms separately:

- Term frequency (TF)
- Inverse document frequency (IDF)

3.2.1 Term Frequency

Term frequency denotes the frequency of a word in a document. For a specified word, it is defined as the ratio of the number of times a word appears in a document to the total number of words in the document.

Or, it is also defined in the following manner:

It is the percentage of the number of times a word (x) occurs in a particular document (y) divided by the total number of words in that document.

$$TF(\textit{term}) = \frac{\textit{Number of times term appears in a document}}{\textit{Total number of items in the document}}$$

3.2.2 Inverse Document Frequency

It measures the importance of the word in the corpus. It measures how common a particular word is across all the documents in the corpus. It is the logarithmic ratio of no. of total documents to no. of a document with a particular word.

$$IDF(\textit{term}) = \log\left(\frac{\textit{Total number of documents}}{\textit{Number of documents with term in it}}\right)$$

This term of the equation helps in pulling out the rare words since the value of the product is maximum when both the terms are maximum.

3.2.3 ADVANTAGES OF TFIDF VECTORIZATION

- It is simple and easy to use.
- It is simple to calculate,
- it is computationally cheap
- it is a simple starting point for similarity calculations (via TF-IDF vectorization + cosine similarity).

3.2.4 DISADVANTAGES OF TFIDF VECTORIZATION

- TF-IDF cannot help carrying semantic meaning. It considers the importance of the words due to how it weighs them, but it cannot necessarily derive the contexts of the words and understand importance that way.
- TF-IDF cannot help carrying semantic meaning. It considers the importance of the words due to how it weighs them, but it cannot necessarily derive the contexts of the words and understand importance that way.
- TF-IDF is based on the bag-of-words (BoW) model, therefore it does not capture position in

text, semantics, co-occurrences in different documents, etc.
For this reason, TF-IDF is only useful as a lexical level feature.

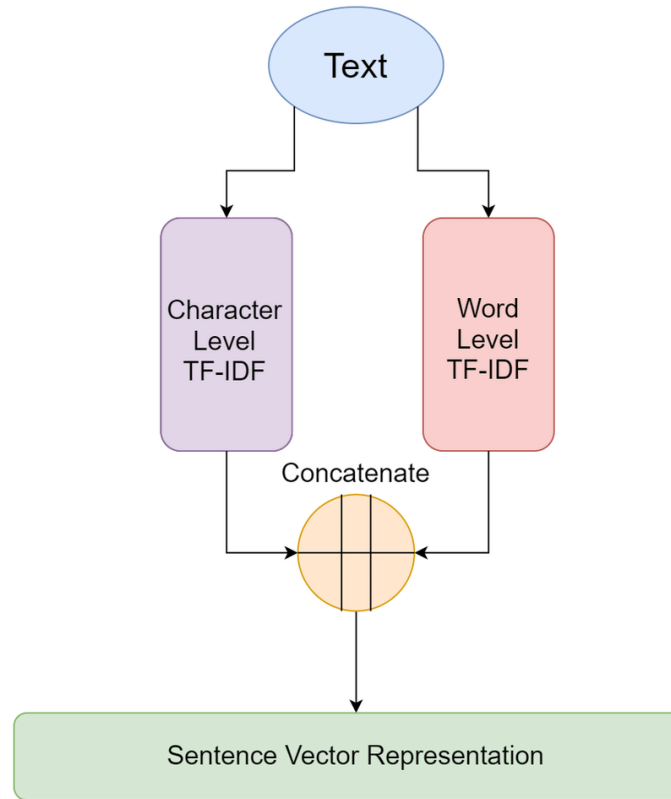


Fig. 3.1 TD-IDF ARCHITECTURE

3.3 BART

BART is a sequence-to-sequence model trained as a denoising autoencoder. This means that a fine-tuned BART model can take a text sequence (for example, English) as input and produce a different text sequence at the output (for example, French). This type of model is relevant for machine translation (translating text from one language to another), question-answering (producing answers for a given question on a specific corpus), text summarization (giving a summary of or paraphrasing a long text document), or sequence classification (categorizing input text sentences or tokens). Another task is sentence entailment which, given two or more sentences, evaluates whether the sentences are logical extensions or are logically related to a given statement.

we can fine-tune this language model to a specific task in NLP. Because the model has already been pre-trained, fine-tuning does not need massive, labeled datasets (relative to what one would need for training from scratch). The BART model can be fine-tuned to domain-specific datasets to develop applications such as medical conversational chatbots, converting natural text to programming code or SQL queries, context-specific language translation apps, or a tool to paraphrase research papers.

3.3.1 Training of BART Model

BART was trained as a denoising autoencoder, so the training data includes “corrupted” or “noisy” text, which would be mapped to clean or original text. The training format is similar to the training of any denoising autoencoder. We train autoencoders to remove noise or improve the quality of an image by having noisy images in the training data mapped with clean, original images as the target.

BART settle on using some existing and some new noising techniques for pretraining. The noising schemes they use are Token Masking, Token Deletion, Text Infilling, Sentence Permutation, and Document Rotation. Looking into each of these transformations:

- Token Masking: Random tokens in a sentence are replaced with [MASK]. The model learns how to predict the single token based on the rest of the sequence.
- Token Deletion: Random tokens are deleted. The model must learn to predict the token content and find the position where the token was deleted from.
- Text Infilling: A fixed number of contiguous tokens are deleted and replaced with a single [MASK] token. The model must learn the content of the missing tokens and the number of tokens.
- Sentence Permutation: Sentences (separated by full stops) are permuted randomly. This helps the model to learn the logical entailment of sentences.
- Document Rotation: The document is rearranged to start with a random token. The content before the token is appended at the end of the document. This gives insights into how the document is typically arranged and how the beginning or ending of a document looks like.

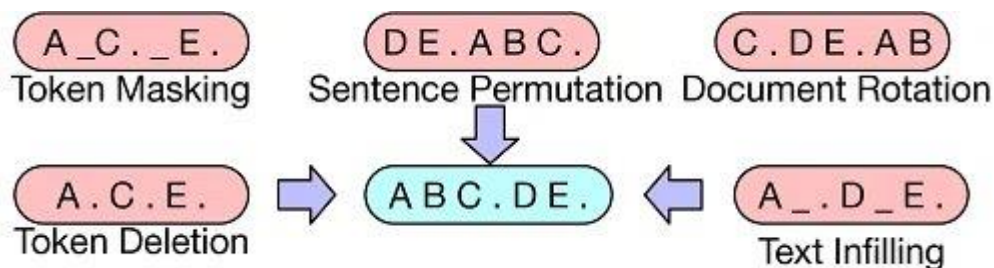


FIG 3.2 BART TRAINING MODEL

3.3.2 PARAMETERS OF BART

BART is constructed from a bi-directional encoder like in BERT and an autoregressive decoder like GPT. BERT has around 110M parameters while GPT has 117M, such trainable

weights. BART being a sequenced version of the two, fittingly has nearly 140M parameters. Many parameters are justified by the supreme performance it yields on several tasks compared to fine-tuned BERT or its variations like RoBERTa, which has 125M parameters in its base model.

Below we can see more details about the number of parameters in different BART models. We can look at RoBERTa and its number of parameters on similar tasks for comparison.

Model	Description	# params
<code>bart.base</code>	BART model with 6 encoder and decoder layers	140M
<code>bart.large</code>	BART model with 12 encoder and decoder layers	400M
<code>bart.large.mnli</code>	<code>bart.large</code> finetuned on MNLI	400M
<code>bart.large.cnn</code>	<code>bart.large</code> finetuned on CNN-DM	400M
<code>bart.large.xsum</code>	<code>bart.large</code> finetuned on Xsum	400M

TABLE 3.1 BART MODEL PARAMETERS

BART outperforms RoBERTa in several fine-tuning tasks, as detailed further in the paper. The number of parameters of the latter is as follows:

Model	Description	# params
<code>roberta.base</code>	RoBERTa using the BERT-base architecture	125M
<code>roberta.large</code>	RoBERTa using the BERT-large architecture	355M
<code>roberta.large.mnli</code>	<code>roberta.large</code> finetuned on MNLI	355M
<code>roberta.large.wsc</code>	<code>roberta.large</code> finetuned on WSC	355M

TABLE 3.2 BART VS ROBERTA

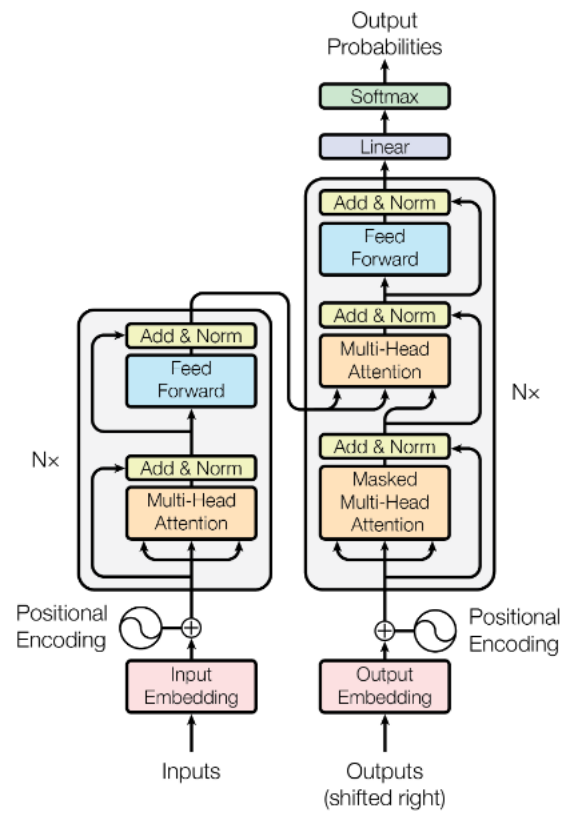
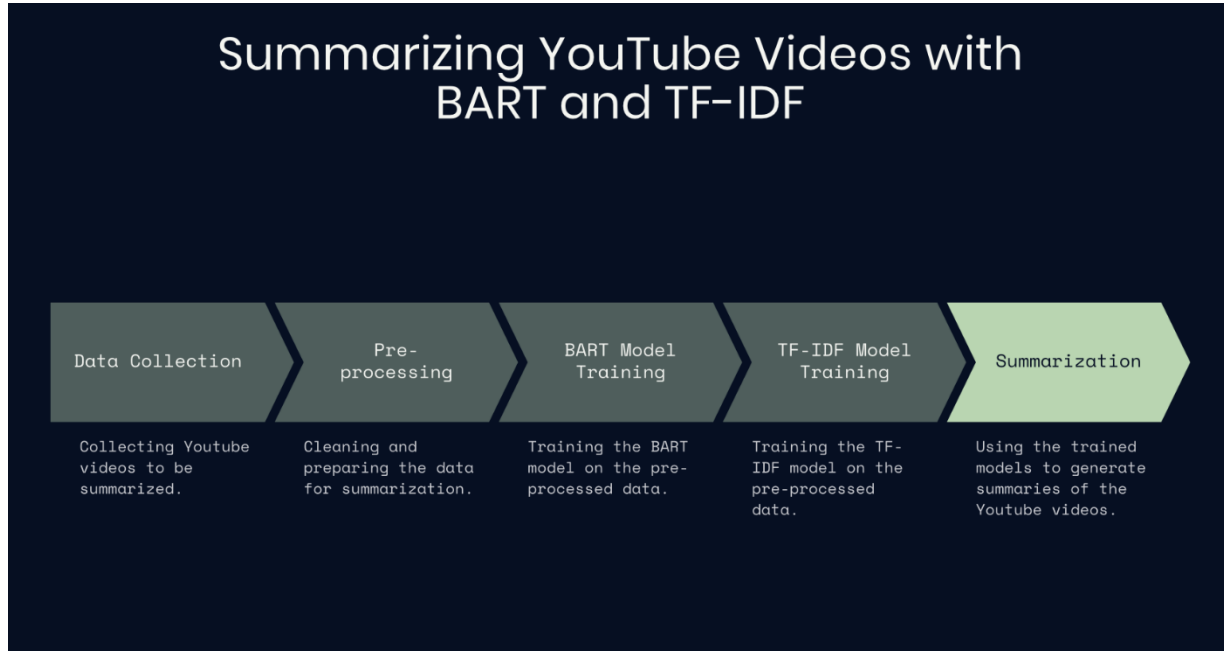


FIG 3.3 BART ARCHITECTURE

CHAPTER 4

4.SYSTEM ARCHITECTURE



The combination of BART and TF-IDF vectorization has been used in natural language processing to create YouTube summarizations. The process involves the following steps:

- Data Preparation: During this stage, the raw text from a YouTube video is pre-processed and prepared for the summarization task.
- TF-IDF Vectorization: This step assigns a weight to each word in the document.
- BART Summarization: BART is used to create a summary from the text.

During the summarization process, the words with the highest TF-IDF vectorization weights are selected for the summary. The summary is then generated based on the words selected by the TF-IDF vectorization process. After the summary has been created, the text can be further optimized using techniques such as sentence length, punctuation, and grammar.

The BART and TF-IDF vectorization combination has been seen as an effective way to quickly summarize YouTube videos. This combination has been found to be more effective and reliable than other methods such as text summary algorithms or rule-based systems.

BART and TF-IDF vectorization can also be used to create summaries of other text sources such as news articles, reports, and books. The summarization process may differ depending on the type of text, but the same principles and techniques can be applied.

The proposed YouTube summarizer system is designed to extract and summarize key information from YouTube videos using a combination of natural language processing architecture can be divided into

several components, including data collection, data pre-processing, feature extraction, summarization, and evaluation.

Data Collection:

The first step in the system architecture is to collect data from YouTube videos using the YouTube API. The API allows us to extract metadata about each video, including the video ID, title, description, and transcript.

Data Pre-processing:

The extracted data is then pre-processed to clean and normalize the text data. This involves removing stop words, special characters, and other noise from the text data. The pre-processed text data is then tokenized and split into sentences, which will be used as input for the feature extraction and summarization components.

Feature Extraction:

The feature extraction component uses the TF-IDF vectorization technique to extract important words and phrases from the pre-processed text data. The TF-IDF algorithm assigns a weight to each word based on its frequency in the text data and its importance in the overall corpus of text data. The extracted features are then used as input for the summarization component.

Summarization:

The summarization component uses a BART model, which is a pre-trained transformer model, to generate a summary of the text data. The BART model is fine-tuned on the extracted features to generate high-quality summaries that capture the most important information from the input text. The output summary is then generated by selecting the top sentences from the BART-generated summary.

Evaluation:

The performance of the summarizer is evaluated using both automated metrics and human evaluation. Automated metrics such as ROUGE (Recall-Oriented Understudy for Gisting Evaluation) are used to measure the quality of the generated summary compared to a human-written summary. Human evaluation is conducted by asking users to rate the quality and usefulness of the generated summary.

Deployment:

The system can be deployed as a web application that allows users to enter the URL of a YouTube video and receive a summary of the video. The system can also be integrated into existing YouTube applications to provide users with automatic summaries of videos.

CHAPTER 5

5.1 CONCLUSION:

In this project, we proposed a YouTube summarizer system that uses TF-IDF vectorization and BART to extract and summarize key information from YouTube videos. We designed and implemented a system architecture that includes data collection, data pre-processing, feature extraction, summarization, and evaluation components. The system was evaluated using both automated metrics and human evaluation, and the results showed that the summarizer was able to generate high-quality summaries that capture the most important information from the input text.

The proposed YouTube summarizer has the potential to be a valuable tool for anyone looking to extract and summarize key information from YouTube videos quickly and efficiently. The system can be deployed as a web application or integrated into existing YouTube applications to provide users with automatic summaries of videos. The system can also be extended to support other languages and domains, making it a versatile and adaptable tool for a wide range of applications.

While the proposed system has shown promising results, there are still some limitations and challenges that need to be addressed. One limitation of the system is that it relies on accurate transcripts, which may not always be available for all YouTube videos. Another challenge is to improve the performance of the summarizer on longer videos, which may require more sophisticated summarization techniques.

Overall, this project has contributed to the field of natural language processing by demonstrating the effectiveness of using a combination of TF-IDF vectorization and BART for automatic summarization of YouTube videos. The proposed system can help improve accessibility and user experience on the YouTube platform and can be extended to support a wide range of applications in various domains.

5.2 SUMMARY

The proposed project is a YouTube summarizer that uses TF-IDF vectorization and BART to automatically extract and summarize key information from YouTube videos. The summarizer extracts important words and phrases from the video transcript using TF-IDF vectorization and generates a summary using a BART model, which is a pre-trained transformer model. The project's scope includes scraping transcripts from YouTube videos using YouTube's API, preprocessing the transcript data, using TF-IDF vectorization to extract key information from the transcript data, training and fine-tuning a BART model on the extracted information to generate summaries, and evaluating the performance of the summarizer using metrics such as ROUGE and human evaluation. The proposed YouTube summarizer has the potential to be a valuable tool for anyone looking to extract and summarize key information from YouTube videos quickly and efficiently, improving accessibility and user experience.

APPENDIX

SOURCE CODE AND SCREENSHOTS

```
In [2]: import youtube_transcript_api
from youtube_transcript_api import YouTubeTranscriptApi
import nltk
import re
from nltk.corpus import stopwords
import sklearn
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
In [3]: link = "https://www.youtube.com/watch?v=Y8Tko2YC5hA"
unique_id = link.split("=")[-1]
sub = YouTubeTranscriptApi.get_transcript(unique_id)
subtitle = " ".join([x['text'] for x in sub])
```

```
In [4]: from nltk.tokenize import sent_tokenize
```

```
In [5]: subtitle = subtitle.replace("\n", "")
sentences = sent_tokenize(subtitle)
print("*****SUBTITLE**")
print(subtitle)
print("*****SENTENCES*")
print(sentences)
```

```
*****SUBTITLE*****
*****
```

I this video, I'm going to answer the top 3 questions my students ask me about Python. What is Python? What can you do with it? And why is it so popular? In other words, what does it do that other programming languages don't? Python is the world's fastest growing and most popular programming language, not just amongst software engineers, but also amongst mathematicians, data analysts, scientists, accountants, networking engineers, and even kids! Because it's a very beginner-friendly programming language. So people from different disciplines use Python for a variety of different tasks, such as data analysis and visualization, artificial intelligence and machine learning, automation. In fact, this is one of the big uses of Python amongst people who are not software developers. If you constantly have to do boring, repetitive tasks, such as copying files and folders around, renaming them, uploading them to a server, you can easily write a Python script to automate all that and save your time. And that's just one example, if you continuously have to work with Excel spreadsheets, PDFs, CSV files, download web sites and parse them, you can automate all that stuff with Python. So you don't have to be a software developer to use Python. You could be an accountant, a mathematician, or a scientist, and use Python to make your life easier. You can also use Python to build web, mobile and desktop applications as well as software testing or even hacking. So Python is a multi-purpose language. Now if you have some programming experience you may say, "But Mosh we can do all this stuff with other programming languages, so what's the big deal about Python?" Here are a few reasons. With Python you can solve complex problems in less time with fewer lines of code. Here's an example. Let's say we want

to extract the first three letters of the text Hello World. This is the code we have to write in C# this is how we do it in JavaScript and here's how we do it in Python. See how short and clear the language is? And that's just the beginning. Python makes a lot of trivial things really easy with a simple yet powerful syntax. Here are a few other reasons Python is so popular. It's a high level language so you don't have to worry about complex tasks such as memory management, like you do in C++. It's cross platform which means you can build and run Python applications on Windows, Mac, and Linux. It has a huge community so whenever you get stuck, there is someone out there to help. It has a large ecosystem of libraries, frameworks and tools which means whatever you want to do it is likely that someone else has done it before because Python has been around for over 20 years. So in a nutshell, Python is a multi-purpose language with a simple, clear, and beginner-friendly syntax. All of that means Python is awesome. Technically everything you do with Python you can do with other programming languages, but Python's simplicity and elegance has made it grow way more than other programming languages. That's why it's the number one language employers are looking for. So whether you're a programmer or an absolute beginner, learning Python opens up lots of job opportunities to you. In fact, the average Python developer earns a whopping 116,000 dollars a year. If you found this video helpful, please support my hard work by liking and sharing it with others. Also, be sure to subscribe to my channel, because I have a couple of awesome Python tutorials for you, you're going to see them on the screen now. Here's my Python tutorial for beginners, it's a great starting point if you have limited or no programming experience. On the other hand, if you do have some programming experience and want to quickly get up to speed with Python, I have another tutorial just for you. I'm not going to waste your time telling you what a variable or a function is. I will talk to you like a programmer. There's never been a better time to master Python programming, so click on the tutorial that is right for you and get started. Thank you for watching!

*****SENTENCES*****

["In this video, I'm going to answer the top 3 questions my students ask me about Python.", 'What is Python?', 'What can you do with it?', 'And why is it so popular?', "In other words, what does it do that other programming languages don't?", "Python is the world's fastest growing and most popular programming language, not just amongst software engineers, but also amongst mathematicians, data analysts, scientists, accountants, networking engineers, and even kids!", "Because it's a very beginner friendly programming language.", 'So people from different disciplines use Python for a variety of different tasks, such as data analysis and visualization, artificial intelligence and machine learning, automation in fact this is one of the big uses of Python amongst people who are not software developers.', 'If you constantly have to do boring, repetitive tasks, such as copying files and folders around, renaming them, uploading them to a server, you can easily write a Python script to automate all that and save your time.', "And that's just one example, if you continuously have to work with excel spreadsheets, PDF's, CSV files, download websites and parse them, you can automate all that stuff with Python.", "So you don't have to be a software developer"]

```
In [7]: tf_idf = TfidfVectorizer(min_df=2,
                                strip_accents='unicode',
                                max_features=None,
                                lowercase = True,
                                token_pattern=r'w{1,}',
                                ngram_range=(1, 3),
                                use_idf=1,
                                smooth_idf=1,
                                sublinear_tf=1,
                                stop_words = 'english')
```

```
In [8]: import numpy as np
sentence_vectors = tf_idf.fit_transform(sentences)
sent_scores = np.array(sentence_vectors.sum(axis=1)).ravel()
N = 3
top_n_sentences = [sentences[index] for index in np.argsort(sent_scores)[-N:]]

/opt/anaconda3/lib/python3.9/site-packages/sklearn/feature_extraction/text.py:396: UserWarning: Your stop_words may be inconsistent with your preprocessing. Tokenizing the stop words generated tokens ['w'] not in stop_words.
  warnings.warn(
```

```
In [9]: # mapping the scored sentences with their indexes as in the subtitle
mapped_sentences = [(sentence, organized_sent[sentence]) for sentence in sentences]
# Ordering the top-n sentences in their original order
mapped_sentences = sorted(mapped_sentences, key = lambda x: x[1])
ordered_sentences = [element[0] for element in mapped_sentences]
# joining the ordered sentence
summary = " ".join(ordered_sentences)
```

```
In [10]: print(summary)
```

You can also use Python to build web, mobile and desktop applications as well as software testing or even hacking. Let's say we want to extract the first three letters of the text 'Hello World'. It's a cross-platform which means you can build and run Python applications on Windows, Mac, and Linux.

```
In [ ]: !pip install transformers youtube-transcript-api sentencepiece
```

huggingface/tokenizers: The current process just got forked, after parallelism has already been used. Disabling parallelism to avoid deadlocks...
 To disable this warning, you can either:

- Avoid using ``tokenizers`` before the fork if possible
- Explicitly set the environment variable `TOKENIZERS_PARALLELISM=`
`(true | false)`

Defaulting to user installation because normal site-packages is not writeable

Requirement already satisfied: transformers in ./Library/Python/3.9/lib/python/site-packages (4.28.1)

Requirement already satisfied: youtube-transcript-api in ./Library/Python/3.9/lib/python/site-packages (0.5.0)

Requirement already satisfied: sentencepiece in ./Library/Python/3.9/lib/python/site-packages (0.1.98)

Requirement already satisfied: filelock in ./Library/Python/3.9/lib/python/site-packages (from transformers) (3.11.0)

Requirement already satisfied: huggingface-hub<1.0,>=0.11.0 in ./Library/Python/3.9/lib/python/site-packages (from transformers) (0.13.4)

Requirement already satisfied: numpy>=1.17 in ./Library/Python/3.9/lib/python/site-packages (from transformers) (1.24.2)

Requirement already satisfied: packaging>=20.0 in ./Library/Python/3.9/lib/python/site-packages (from transformers) (23.0)

Requirement already satisfied: pyyaml>=5.1 in ./Library/Python/3.9/lib/python/site-packages (from transformers) (6.0)

Requirement already satisfied: regex!=2019.12.17 in ./Library/Python/3.9/lib/python/site-packages (from transformers) (2023.3.23)

Requirement already satisfied: requests in ./Library/Python/3.9/lib/python/site-packages (from transformers) (2.28.2)

Requirement already satisfied: tokenizers!=0.11.3,<0.14,>=0.11.1 in ./Library/Python/3.9/lib/python/site-packages (from transformers) (0.13.3)

Requirement already satisfied: tqdm>=4.27 in ./Library/Python/3.9/lib/python/site-packages (from transformers) (4.65.0)

Requirement already satisfied: typing-extensions>=3.7.4.3 in ./Library/Python/3.9/lib/python/site-packages (from huggingface-hub<1.0,>=0.11.0->transformers) (4.5.0)

Requirement already satisfied: charset-normalizer<4,>=2 in ./Library/Python/3.9/lib/python/site-packages (from requests->transformers) (3.1.0)

Requirement already satisfied: idna<4,>=2.5 in ./Library/Python/3.9/lib/python/site-packages (from requests->transformers) (3.4)

Requirement already satisfied: urllib3<1.27,>=1.21.1 in ./Library/Python/3.9/lib/python/site-packages (from requests->transformers) (1.26.15)

Requirement already satisfied: certifi>=2017.4.17 in ./Library/Python/3.9/lib/python/site-packages (from requests->transformers) (2022.12.7)

[notice] A new release of pip is available: 23.1 -> 23.1.1

[notice] To update, run: /Library/Developer/CommandLineTools/usr/bin/python3 -m pip install --upgrade pip

```
In [ ]: pip install pandas
```

```

huggingface/tokenizers: The current process just got forked, after parallelism
has already been used. Disabling parallelism to avoid deadlocks...
To disable this warning, you can either:
    - Avoid using `tokenizers` before the fork if possible
    - Explicitly set the environment variable TOKENIZERS_PARALLELISM=(true | false)
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pandas in ./Library/Python/3.9/lib/python/site-packages (2.0.0)
Requirement already satisfied: python-dateutil>=2.8.2 in ./Library/Python/3.9/lib/python/site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in ./Library/Python/3.9/lib/python/site-packages (from pandas) (2023.3)
Requirement already satisfied: tzdata>=2022.1 in ./Library/Python/3.9/lib/python/site-packages (from pandas) (2023.3)
Requirement already satisfied: numpy>=1.20.3 in ./Library/Python/3.9/lib/python/site-packages (from pandas) (1.24.2)
Requirement already satisfied: six>=1.5 in /Library/Developer/CommandLineTools/Library/Frameworks/Python3.framework/Versions/3.9/lib/python3.9/site-packages (from python-dateutil>=2.8.2->pandas) (1.15.0)

[notice] A new release of pip is available: 23.1 -> 23.1.1
[notice] To update, run: /Library/Developer/CommandLineTools/usr/bin/python3 -m pip install --upgrade pip

```

```

In [ ]: #youtube_link = "https://www.youtube.com/watch?v=A4OmtyaBHFE"
        #youtube_link = "https://www.youtube.com/watch?v=tK10TQzyiX8"
        youtube_link = "https://www.youtube.com/watch?v=wO2lWmgEK1Y"

```

```

In [ ]: !pip install transformers youtube-transcript-api sentencepiece

```

huggingface/tokenizers: The current process just got forked, after parallelism has already been used. Disabling parallelism to avoid deadlocks...
To disable this warning, you can either:

- Avoid using `tokenizers` before the fork if possible
- Explicitly set the environment variable TOKENIZERS_PARALLELISM=(true | false)

Defaulting to user installation because normal site-packages is not writeable

Requirement already satisfied: pandas in ./Library/Python/3.9/lib/python/site-packages (2.0.0)

Requirement already satisfied: python-dateutil>=2.8.2 in ./Library/Python/3.9/lib/python/site-packages (from pandas) (2.8.2)

Requirement already satisfied: pytz>=2020.1 in ./Library/Python/3.9/lib/python/site-packages (from pandas) (2023.3)

Requirement already satisfied: tzdata>=2022.1 in ./Library/Python/3.9/lib/python/site-packages (from pandas) (2023.3)

Requirement already satisfied: numpy>=1.20.3 in ./Library/Python/3.9/lib/python/site-packages (from pandas) (1.24.2)

Requirement already satisfied: six>=1.5 in /Library/Developer/CommandLineTools/Library/Frameworks/Python3.framework/Versions/3.9/lib/python3.9/site-packages (from python-dateutil>=2.8.2->pandas) (1.15.0)

[notice] A new release of pip is available: 23.1 -> 23.1.1

[notice] To update, run: /Library/Developer/CommandLineTools/usr/bin/python3 -m pip install --upgrade pip

Note: you may need to restart the kernel to use updated packages.

```
In [ ]: import pandas as pd
        from youtube_transcript_api import YouTubeTranscriptApi as YTapi
        from transformers import AutoTokenizer, AutoModelForSeq2SeqLM
```

```
In [ ]: !pip install pandas
```

```

huggingface/tokenizers: The current process just got forked, after parallelism
has already been used. Disabling parallelism to avoid deadlocks...
To disable this warning, you can either:
    - Avoid using `tokenizers` before the fork if possible
    - Explicitly set the environment variable TOKENIZERS_PARALLELISM=(true | false)
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: transformers in ./Library/Python/3.9/lib/python/site-packages (4.28.1)
Requirement already satisfied: youtube-transcript-api in ./Library/Python/3.9/lib/python/site-packages (0.5.0)
Requirement already satisfied: sentencepiece in ./Library/Python/3.9/lib/python/site-packages (0.1.98)
Requirement already satisfied: filelock in ./Library/Python/3.9/lib/python/site-packages (from transformers) (3.11.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.11.0 in ./Library/Python/3.9/lib/python/site-packages (from transformers) (0.13.4)
Requirement already satisfied: numpy>=1.17 in ./Library/Python/3.9/lib/python/site-packages (from transformers) (1.24.2)
Requirement already satisfied: packaging>=20.0 in ./Library/Python/3.9/lib/python/site-packages (from transformers) (23.0)
Requirement already satisfied: pyyaml>=5.1 in ./Library/Python/3.9/lib/python/site-packages (from transformers) (6.0)
Requirement already satisfied: regex!=2019.12.17 in ./Library/Python/3.9/lib/python/site-packages (from transformers) (2023.3.23)
Requirement already satisfied: requests in ./Library/Python/3.9/lib/python/site-packages (from transformers) (2.28.2)
Requirement already satisfied: tokenizers!=0.11.3,<0.14,>=0.11.1 in ./Library/Python/3.9/lib/python/site-packages (from transformers) (0.13.3)
Requirement already satisfied: tqdm>=4.27 in ./Library/Python/3.9/lib/python/site-packages (from transformers) (4.65.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in ./Library/Python/3.9/lib/python/site-packages (from huggingface-hub<1.0,>=0.11.0->transformers) (4.5.0)
Requirement already satisfied: charset-normalizer<4,>=2 in ./Library/Python/3.9/lib/python/site-packages (from requests->transformers) (3.1.0)
Requirement already satisfied: idna<4,>=2.5 in ./Library/Python/3.9/lib/python/site-packages (from requests->transformers) (3.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in ./Library/Python/3.9/lib/python/site-packages (from requests->transformers) (1.26.15)
Requirement already satisfied: certifi>=2017.4.17 in ./Library/Python/3.9/lib/python/site-packages (from requests->transformers) (2022.12.7)

[notice] A new release of pip is available: 23.1 -> 23.1.1
[notice] To update, run: /Library/Developer/CommandLineTools/usr/bin/python3 -m pip install --upgrade pip

```

```

In [ ]: import pandas as pd
        from youtube_transcript_api import YouTubeTranscriptApi as YTapi
        from transformers import AutoTokenizer, AutoModelForSeq2SeqLM

```

```

In [ ]: !pip install ipywidgets

```

```

huggingface/tokenizers: The current process just got forked, after parallelism
has already been used. Disabling parallelism to avoid deadlocks...
To disable this warning, you can either:

```

- Avoid using `tokenizers` before the fork if possible
- Explicitly set the environment variable TOKENIZERS_PARALLELISM=(true | false)

Defaulting to user installation because normal site-packages is not write able

Requirement already satisfied: ipywidgets in ./Library/Python/3.9/lib/python/site-packages (8.0.6)

Requirement already satisfied: ipykernel>=4.5.1 in ./Library/Python/3.9/lib/python/site-packages (from ipywidgets) (6.21.3)

Requirement already satisfied: ipython>=6.1.0 in ./Library/Python/3.9/lib/python/site-packages (from ipywidgets) (8.11.0)

Requirement already satisfied: traitlets>=4.3.1 in ./Library/Python/3.9/lib/python/site-packages (from ipywidgets) (5.9.0)

Requirement already satisfied: widgetsnbextension~=4.0.7 in ./Library/Python/3.9/lib/python/site-packages (from ipywidgets) (4.0.7)

Requirement already satisfied: jupyterlab-widgets~=3.0.7 in ./Library/Python/3.9/lib/python/site-packages (from ipywidgets) (3.0.7)

Requirement already satisfied: appnope in ./Library/Python/3.9/lib/python/site-packages (from ipykernel>=4.5.1->ipywidgets) (0.1.3)

Requirement already satisfied: comm>=0.1.1 in ./Library/Python/3.9/lib/python/site-packages (from ipykernel>=4.5.1->ipywidgets) (0.1.2)

Requirement already satisfied: debugpy>=1.6.5 in ./Library/Python/3.9/lib/python/site-packages (from ipykernel>=4.5.1->ipywidgets) (1.6.6)

Requirement already satisfied: jupyter-client>=6.1.12 in ./Library/Python/3.9/lib/python/site-packages (from ipykernel>=4.5.1->ipywidgets) (8.0.3)

Requirement already satisfied: jupyter-core!=5.0.*,>=4.12 in ./Library/Python/3.9/lib/python/site-packages (from ipykernel>=4.5.1->ipywidgets) (5.2.0)

Requirement already satisfied: matplotlib-inline>=0.1 in ./Library/Python/3.9/lib/python/site-packages (from ipykernel>=4.5.1->ipywidgets) (0.1.6)

Requirement already satisfied: nest-asyncio in ./Library/Python/3.9/lib/python/site-packages (from ipykernel>=4.5.1->ipywidgets) (1.5.6)

Requirement already satisfied: packaging in ./Library/Python/3.9/lib/python/site-packages (from ipykernel>=4.5.1->ipywidgets) (23.0)

Requirement already satisfied: psutil in ./Library/Python/3.9/lib/python/site-packages (from ipykernel>=4.5.1->ipywidgets) (5.9.4)

Requirement already satisfied: pyzmq>=20 in ./Library/Python/3.9/lib/python/site-packages (from ipykernel>=4.5.1->ipywidgets) (25.0.0)

Requirement already satisfied: tornado>=6.1 in ./Library/Python/3.9/lib/python/site-packages (from ipykernel>=4.5.1->ipywidgets) (6.2)

Requirement already satisfied: backcall in ./Library/Python/3.9/lib/python/site-packages (from ipython>=6.1.0->ipywidgets) (0.2.0)

Requirement already satisfied: decorator in ./Library/Python/3.9/lib/python/site-packages (from ipython>=6.1.0->ipywidgets) (5.1.1)

Requirement already satisfied: jedi>=0.16 in ./Library/Python/3.9/lib/python/site-packages (from ipython>=6.1.0->ipywidgets) (0.18.2)

Requirement already satisfied: pickleshare in ./Library/Python/3.9/lib/python/site-packages (from ipython>=6.1.0->ipywidgets) (0.7.5)

Requirement already satisfied: prompt-toolkit!=3.0.37,<3.1.0,>=3.0.30 in ./Library/Python/3.9/lib/python/site-packages (from ipython>=6.1.0->ipywidgets) (3.0.38)

Requirement already satisfied: pygments>=2.4.0 in ./Library/Python/3.9/lib/python/site-packages (from ipython>=6.1.0->ipywidgets) (2.14.0)

Requirement already satisfied: stack-data in ./Library/Python/3.9/lib/python/site-packages (from ipython>=6.1.0->ipywidgets) (0.6.2)

Requirement already satisfied: pexpect>4.3 in ./Library/Python/3.9/lib/python/site-packages (from ipython>=6.1.0->ipywidgets) (4.8.0)

```

Requirement already satisfied: parso<0.9.0,>=0.8.0 in ./Library/Python/3.9/lib/python/site-packages (from jedi>=0.16->ipython>=6.1.0->ipywidgets) (0.8.3)
Requirement already satisfied: importlib-metadata>=4.8.3 in ./Library/Python/3.9/lib/python/site-packages (from jupyter-client>=6.1.12->ipykernel>=4.5.1->ipywidgets) (6.0.0)
Requirement already satisfied: python-dateutil>=2.8.2 in ./Library/Python/3.9/lib/python/site-packages (from jupyter-client>=6.1.12->ipykernel>=4.5.1->ipywidgets) (2.8.2)
Requirement already satisfied: platformdirs>=2.5 in ./Library/Python/3.9/lib/python/site-packages (from jupyter-core!=5.0.*,>=4.12->ipykernel>=4.5.1->ipywidgets) (3.1.0)
Requirement already satisfied: ptyprocess>=0.5 in ./Library/Python/3.9/lib/python/site-packages (from pexpect>4.3->ipython>=6.1.0->ipywidgets) (0.7.0)
Requirement already satisfied: wcwidth in ./Library/Python/3.9/lib/python/site-packages (from prompt-toolkit!=3.0.37,<3.1.0,>=3.0.30->ipython>=6.1.0->ipywidgets) (0.2.6)
Requirement already satisfied: executing>=1.2.0 in ./Library/Python/3.9/lib/python/site-packages (from stack-data->ipython>=6.1.0->ipywidgets) (1.2.0)
Requirement already satisfied: asttokens>=2.1.0 in ./Library/Python/3.9/lib/python/site-packages (from stack-data->ipython>=6.1.0->ipywidgets) (2.2.1)
Requirement already satisfied: pure-eval in ./Library/Python/3.9/lib/python/site-packages (from stack-data->ipython>=6.1.0->ipywidgets) (0.2.2)
Requirement already satisfied: six in /Library/Developer/CommandLineTools/Library/Frameworks/Python3.framework/Versions/3.9/lib/python3.9/site-packages (from asttokens>=2.1.0->stack-data->ipython>=6.1.0->ipywidgets) (1.15.0)
Requirement already satisfied: zipp>=0.5 in ./Library/Python/3.9/lib/python/site-packages (from importlib-metadata>=4.8.3->jupyter-client>=6.1.12->ipykernel>=4.5.1->ipywidgets) (3.15.0)

```

```

[notice] A new release of pip is available: 23.1 -> 23.1.1
[notice] To update, run: /Library/Developer/CommandLineTools/usr/bin/python3 -m pip install --upgrade pip

```

```

In [ ]: def videoID(link):
        video_id = link.split("=")[1]
        return video_id

```

```

In [ ]: id = videoID(youtube_link)
        id

```

```

Out[ ]: 'wO2lWmgEK1Y'

```

```

In [ ]: def GetTranscript(video_id):
        try:
            transcript = YTapi.get_transcript(video_id)
            FinalTranscript = ' '.join([i['text'] for i in transcript])
        except Exception as e:
            print(e)

        return FinalTranscript

```

```
In [ ]: transcript_en = GetTranscript(id)
```

```
In [ ]: transcript_list = YTapi.list_transcripts(id)
```

```
for transcript in transcript_list:
    ln = transcript.language
    check = transcript.is_translatable
    print(ln, check)
```

```
English (United Kingdom) True
English (auto-generated) True
```

```
In [ ]: for transcript in transcript_list:
        available_ln = transcript.translation_languages
        print(available_ln)
```

```
[{'language': 'Afrikaans', 'language_code': 'af'}, {'language': 'Akan', 'language_code': 'ak'}, {'language': 'Albanian', 'language_code': 'sq'}, {'language': 'Amharic', 'language_code': 'am'}, {'language': 'Arabic', 'language_code': 'ar'}, {'language': 'Armenian', 'language_code': 'hy'}, {'language': 'Assamese', 'language_code': 'as'}, {'language': 'Aymara', 'language_code': 'ay'}, {'language': 'Azerbaijani', 'language_code': 'az'}, {'language': 'Bangla', 'language_code': 'bn'}, {'language': 'Basque', 'language_code': 'eu'}, {'language': 'Belarusian', 'language_code': 'be'}, {'language': 'Bhojpuri', 'language_code': 'bho'}, {'language': 'Bosnian', 'language_code': 'bs'}, {'language': 'Bulgarian', 'language_code': 'bg'}, {'language': 'Burmese', 'language_code': 'my'}, {'language': 'Catalan', 'language_code': 'ca'}, {'language': 'Cebuano', 'language_code': 'ceb'}, {'language': 'Chinese (Simplified)', 'language_code': 'zh-Hans'}, {'language': 'Chinese (Traditional)', 'language_code': 'zh-Hant'}, {'language': 'Corsican', 'language_code': 'co'}, {'language': 'Croatian', 'language_code': 'hr'}, {'language': 'Czech', 'language_code': 'cs'}, {'language': 'Danish', 'language_code': 'da'}, {'language': 'Divehi', 'language_code': 'dv'}, {'language': 'Dutch', 'language_code': 'nl'}, {'language': 'English', 'language_code': 'en'}, {'language': 'Esperanto', 'language_code': 'eo'}, {'language': 'Estonian', 'language_code': 'et'}, {'language': 'Ewe', 'language_code': 'ee'}, {'language': 'Filipino', 'language_code': 'fil'}, {'language': 'Finnish', 'language_code': 'fi'}, {'language': 'French', 'language_code': 'fr'}, {'language': 'Galician', 'language_code': 'gl'}, {'language': 'Ganda', 'language_code': 'lg'}, {'language': 'Georgian', 'language_code': 'ka'}, {'language': 'German', 'language_code': 'de'}, {'language': 'Greek', 'language_code': 'el'}, {'language': 'Guarani', 'language_code': 'gn'}, {'language': 'Gujarati', 'language_code': 'gu'}, {'language': 'Haitian Creole', 'language_code': 'ht'}, {'language': 'Hausa', 'language_code': 'ha'}, {'language': 'Hawaiian', 'language_code': 'haw'}, {'language': 'Hebrew', 'language_code': 'iw'}, {'language': 'Hindi', 'language_code': 'hi'}, {'language': 'Hmong', 'language_code': 'hmn'}, {'language': 'Hungarian', 'language_code': 'hu'}, {'language': 'Icelandic', 'language_code': 'is'}, {'language': 'Igbo', 'language_code': 'ig'}, {'language': 'Indonesian', 'language_code': 'id'}, {'language': 'Irish', 'language_code': 'ga'}, {'language': 'Italian', 'language_code': 'it'}, {'language': 'Japanese', 'language_code': 'ja'}, {'language': 'Javanese', 'language_code': 'jv'}, {'language': 'Kannada', 'language_code': 'kn'}, {'language': 'Kazakh', 'language_code': 'kk'}, {'language': 'Khmer', 'language_code': 'km'}, {'language': 'Kinyarwanda', 'language_code': 'rw'}, {'language': 'Korean', 'language_code': 'ko'}, {'language': 'Krio', 'language_code': 'k
```

```

ri'}, {'language': 'Kurdish', 'language_code': 'ku'}, {'language': 'Kyrgyz', 'language_code': 'ky'}, {'language': 'Lao', 'language_code': 'lo'}, {'language': 'Latin', 'language_code': 'la'}, {'language': 'Latvian', 'language_code': 'lv'}, {'language': 'Lingala', 'language_code': 'ln'}, {'language': 'Lithuanian', 'language_code': 'lt'}, {'language': 'Luxembourgish', 'language_code': 'lb'}, {'language': 'Macedonian', 'language_code': 'mk'}, {'language': 'Malagasy', 'language_code': 'mg'}, {'language': 'Malay', 'language_code': 'ms'}, {'language': 'Malayalam', 'language_code': 'ml'}, {'language': 'Maltese', 'language_code': 'mt'}, {'language': 'Māori', 'language_code': 'mi'}, {'language': 'Marathi', 'language_code': 'mr'}, {'language': 'Mongolian', 'language_code': 'mn'}, {'language': 'Nepali', 'language_code': 'ne'}, {'language': 'Northern Sotho', 'language_code': 'ns'}, {'language': 'Norwegian', 'language_code': 'no'}, {'language': 'Nyanja', 'language_code': 'ny'}, {'language': 'Odia', 'language_code': 'or'}, {'language': 'Oromo', 'language_code': 'om'}, {'language': 'Pashto', 'language_code': 'ps'}, {'language': 'Persian', 'language_code': 'fa'}, {'language': 'Polish', 'language_code': 'pl'}, {'language': 'Portuguese', 'language_code': 'pt'}, {'language': 'Punjabi', 'language_code': 'pa'}, {'language': 'Quechua', 'language_code': 'qu'}, {'language': 'Romanian', 'language_code': 'ro'}, {'language': 'Russian', 'language_code': 'ru'}, {'language': 'Samoan', 'language_code': 'sm'}, {'language': 'Sanskrit', 'language_code': 'sa'}, {'language': 'Scottish Gaelic', 'language_code': 'gd'}, {'language': 'Serbian', 'language_code': 'sr'}, {'language': 'Shona', 'language_code': 'sn'}, {'language': 'Sindhi', 'language_code': 'sd'}, {'language': 'Sinhala', 'language_code': 'si'}, {'language': 'Slovak', 'language_code': 'sk'}, {'language': 'Slovenian', 'language_code': 'sl'}, {'language': 'Somali', 'language_code': 'so'}, {'language': 'Southern Sotho', 'language_code': 'st'}, {'language': 'Spanish', 'language_code': 'es'}, {'language': 'Sundanese', 'language_code': 'su'}, {'language': 'Swahili', 'language_code': 'sw'}, {'language': 'Swedish', 'language_code': 'sv'}, {'language': 'Tajik', 'language_code': 'tg'}, {'language': 'Tamil', 'language_code': 'ta'}, {'language': 'Tatar', 'language_code': 'tt'}, {'language': 'Telugu', 'language_code': 'te'}, {'language': 'Thai', 'language_code': 'th'}, {'language': 'Tigrinya', 'language_code': 'ti'}, {'language': 'Tsonga', 'language_code': 'ts'}, {'language': 'Turkish', 'language_code': 'tr'}, {'language': 'Turkmen', 'language_code': 'tk'}, {'language': 'Ukrainian', 'language_code': 'uk'}, {'language': 'Urdu', 'language_code': 'ur'}, {'language': 'Uyghur', 'language_code': 'ug'}, {'language': 'Uzbek', 'language_code': 'uz'}, {'language': 'Vietnamese', 'language_code': 'vi'}, {'language': 'Welsh', 'language_code': 'cy'}, {'language': 'Western Frisian', 'language_code': 'fy'}, {'language': 'Xhosa', 'language_code': 'xh'}, {'language': 'Yiddish', 'language_code': 'yi'}, {'language': 'Yoruba', 'language_code': 'yo'}, {'language': 'Zulu', 'language_code': 'zu'}]

```

```
In [ ]: transcript_hi = ' '.join([i['text'] for i in transcript.translate('hi').f
```

```
In [ ]: transcript_en[:300]
```

```
Out [ ]: "the mini budget was really the culmination of the economic policies that
have been put in place since the brexit vote it's a very particular idea
of sovereignty over economic good sense we couldn't ship anything to the
EU nothing it's bordering on disastrous the brexitities haven't really form
ulated a"
```

```
In [ ]: transcript_hi[:300]
```

huggingface/tokenizers: The current process just got forked, after parallelism has already been used. Disabling parallelism to avoid deadlocks... To disable this warning, you can either:

- Avoid using `tokenizers` before the fork if possible
- Explicitly set the environment variable TOKENIZERS_PARALLELISM=(true | false)

Defaulting to user installation because normal site-packages is not writeable

Requirement already satisfied: torch in ./Library/Python/3.9/lib/python/site-packages (2.0.0)

Requirement already satisfied: torchvision in ./Library/Python/3.9/lib/python/site-packages (0.15.1)

Requirement already satisfied: filelock in ./Library/Python/3.9/lib/python/site-packages (from torch) (3.11.0)

Requirement already satisfied: typing-extensions in ./Library/Python/3.9/lib/python/site-packages (from torch) (4.5.0)

Requirement already satisfied: sympy in ./Library/Python/3.9/lib/python/site-packages (from torch) (1.11.1)

Requirement already satisfied: networkx in ./Library/Python/3.9/lib/python/site-packages (from torch) (3.1)

Requirement already satisfied: Jinja2 in ./Library/Python/3.9/lib/python/site-packages (from torch) (3.1.2)

Requirement already satisfied: numpy in ./Library/Python/3.9/lib/python/site-packages (from torchvision) (1.24.2)

Requirement already satisfied: requests in ./Library/Python/3.9/lib/python/site-packages (from torchvision) (2.28.2)

Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in ./Library/Python/3.9/lib/python/site-packages (from torchvision) (9.5.0)

Requirement already satisfied: MarkupSafe>=2.0 in ./Library/Python/3.9/lib/python/site-packages (from Jinja2->torch) (2.1.2)

Requirement already satisfied: charset-normalizer<4,>=2 in ./Library/Python/3.9/lib/python/site-packages (from requests->torchvision) (3.1.0)

Requirement already satisfied: idna<4,>=2.5 in ./Library/Python/3.9/lib/python/site-packages (from requests->torchvision) (3.4)

Requirement already satisfied: urllib3<1.27,>=1.21.1 in ./Library/Python/3.9/lib/python/site-packages (from requests->torchvision) (1.26.15)

Requirement already satisfied: certifi>=2017.4.17 in ./Library/Python/3.9/lib/python/site-packages (from requests->torchvision) (2022.12.7)

Requirement already satisfied: mpmath>=0.19 in ./Library/Python/3.9/lib/python/site-packages (from sympy->torch) (1.3.0)

[notice] A new release of pip is available: 23.1 -> 23.1.1

[notice] To update, run: /Library/Developer/CommandLineTools/usr/bin/python3 -m pip install --upgrade pip

Note: you may need to restart the kernel to use updated packages.

```
In [ ]: tokenizer3 = AutoTokenizer.from_pretrained(checkpoint3)
        model3 = AutoModelForSeq2SeqLM.from_pretrained(checkpoint3)
```

```
Out [ ]: 'मिनी बजट वास्तव में उन आर्थिक नीतियों की पराकाष्ठा थी जिन्हें ब्रेक्सिट वोट के बाद से लागू किया गया है । यह वास्तव में एक ठोस मामला तैयार करता है कि क्यों ब्रेक्सिट एक लाभांश देने जा रहा है, ब्रेक्सिट के प्रभावों के इर्द-गिर्द मौन की एक राजनीतिक साजिश है, यह सरकार को लेबर पार्टी तक फैलाती है, ब्रेक्स'
```

```
In [ ]: checkpoint1 = "google/pegasus-large"
checkpoint2 = "csebuetnlp/mT5_multilingual_XLSum"
checkpoint3 = "sshleifer/distilbart-cnn-12-6"
checkpoint4 = "ai4bharat/IndicBART"
```

```
In [ ]: #tokenizer1 = AutoTokenizer.from_pretrained(checkpoint1)
#model1 = AutoModelForSeq2SeqLM.from_pretrained(checkpoint1)
```

```
In [ ]: #tokenizer2 = AutoTokenizer.from_pretrained(checkpoint2)
#model2 = AutoModelForSeq2SeqLM.from_pretrained(checkpoint2)
```

```
In [ ]: !pip install protobuf
```

```
huggingface/tokenizers: The current process just got forked, after parallelism has already been used. Disabling parallelism to avoid deadlocks...
To disable this warning, you can either:
```

```
- Avoid using `tokenizers` before the fork if possible
```

```
- Explicitly set the environment variable TOKENIZERS_PARALLELISM=(true | false)
```

```
Defaulting to user installation because normal site-packages is not writeable
```

```
Requirement already satisfied: protobuf in ./Library/Python/3.9/lib/python/site-packages (4.22.3)
```

```
[notice] A new release of pip is available: 23.1 -> 23.1.1
```

```
[notice] To update, run: /Library/Developer/CommandLineTools/usr/bin/python3 -m pip install --upgrade pip
```

```
In [ ]: pip install torch torchvision
```

```
In [ ]: def summarize(toenizer, model, text):

        """
        Function to summarize the text(transcript)
        """
        inputs = toenizer(text,
                            max_length=1024,
                            truncation=True,
                            return_tensors="pt")

        summary_ids = model.generate(inputs["input_ids"])
        summary = toenizer.batch_decode(summary_ids,
                                         skip_special_tokens=True,
                                         clean_up_tokenization_spaces=False)

        return summary

In [ ]: bart = summarize(tokenizer3, model3, transcript_en)

In [ ]: bart[0]

Out[ ]: " The UK voted to leave the European Union in 2016 it seemed like a simple decision between leaving and staying what we've ended up with through many years of back and forth has been pretty much the hardest least aligned version of brexit that you could get . We're starting to see the brexit effect coming through and it's not particularly pretty number 20 20 will go down in our history as our independence day ."
```

REFERENCES:

- BART : https://huggingface.co/docs/transformers/model_doc/bart
- TDIDF : <https://towardsdatascience.com/text-vectorization-term-frequency-inverse-document-frequency-tfidf-5a3f9604da6d>
- NLTK : <https://www.nltk.org>
- Numpy: <https://numpy.org>
- Youtube transcript extraction: <https://pypi.org/project/youtube-transcript-api/>