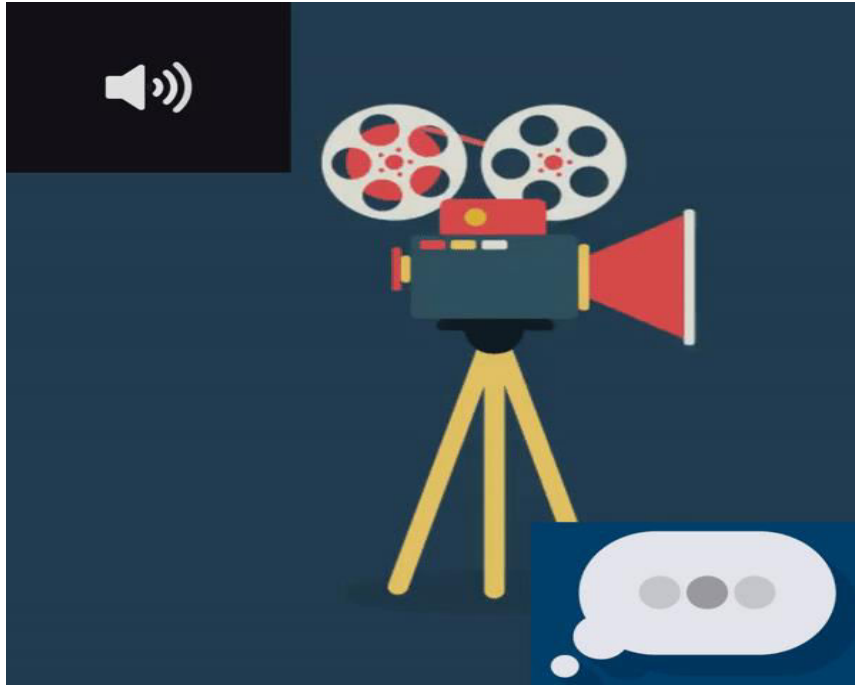


# CIS 8395 - Big Data Analytics Experience (Spring 2023) Project Paper

## Video to Text Summarizer



**Submitted By: Data Pirates**

Harini Prabha Baskar Jayanthi  
Salma Chanbasha Nandikotkur  
Prathyusha Parimi  
Jennifer An

## Table of Contents

<a href="#">3</a> 1. INTRODUCTION .....	3
2. OBJECTIVE .....	3
<a href="#">4</a> 3. BUSINESS PROBLEM .....	4
4. HIGH-LEVEL APPROACH AND SOLUTION .....	4
<a href="#">7</a> 5. ARCHITECTURE .....	7
6. DATA SOURCE .....	8
7. EXTRACT, TRANSFORM AND LOAD (ETL) .....	12
8. DATA CLEANING .....	14
9. DATA STORAGE .....	16
10. DATA LINEAGE .....	18
11. EXPLORATORY DATA ANALYSIS .....	18
12. MODELS .....	20
13.. VISUALIZATION .....	24
14. CONNECTING GOOGLE COLAB TO AN EC2 INSTANCE .....	28
15. CHALLENGES .....	32
16. INDIVIDUAL CONCLUSION AND LEARNING .....	33
17. REFERENCES .....	25

## 1. INTRODUCTION

Video to text summarization is a set of techniques applied in the context of Natural Language Processing (NLP) to shorten the original video transcription in a way that key information is preserved. Video to text summarization is useful in contexts where there is a need for consuming large amounts of video data easier and quicker. Additionally, such applications can be applied in contexts where we need to deal with audio files, which means that the first step would be to perform Speech to Text on the video content prior to Text Summarization and then use that output as input to the service that will be performing the summarization task.



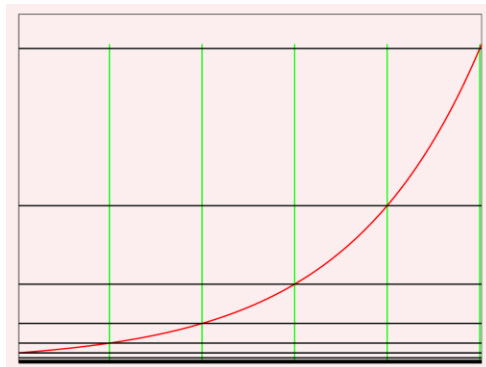
## 2. OBJECTIVE

With the present explosion of data circulating the digital space, which is mostly non-structured textual data, there is a need to develop automatic text summarization tools that allow people to get insights from them easily. Currently, we enjoy quick access to enormous amounts of information. However, most of this information is redundant, insignificant, and may not convey the intended meaning. For example, let's say you are a student or a professional who needs to review a long lecture or a conference video, but you do not have the time to watch the entire video in one sitting.

Therefore, using automatic text summarizers capable of extracting useful information that leaves out inessential and insignificant data is becoming vital. Implementing summarization can enhance the readability of documents, reduce the time spent researching for information, and allow for more information to be fitted in a particular area.

### 3. BUSINESS PROBLEM

In July 2015, YouTube revealed that it receives over 400 hours of video content every single minute, which translates to 65.7 years' worth of content uploaded every day. Since then, we are experiencing an even stronger engagement of consumers with both online video platforms and devices (e.g., smart-phones, wearables etc.) that carry powerful video recording sensors and allow instant uploading of the captured video on the Web. According to newer estimates, YouTube now receives 500 hours of video per minute; and YouTube is just one of the many video hosting platforms that host large volumes of video content. So, how is it possible for someone to efficiently navigate within endless collections of videos, and find the video content that s/he is looking for?



### 4. HIGH-LEVEL APPROACH AND SOLUTION

By using automatic text summarization tools, the key information and main ideas presented in the video can be extracted and presented in a concise and readable format. This can save time and effort for individuals who need to review long video content, as they can quickly skim through the summarized text and identify the relevant information without having to watch the entire video. Additionally, the summarized text can be easily searched and organized, allowing for more efficient management and access to large volumes of video content. By implementing video to text summarization, businesses and individuals can

increase productivity, save time, and efficiently utilize the vast amount of video content available on the internet. For video to text summarization there is essentially two methods:


- **Extractive Summarization:** In extraction-based summarization, a subset of words that represent the most important points is pulled from a piece of text and combined to make a summary. Think of it as a highlighter—which selects the main information from a source text.



Highlighter = Extractive-based summarization

In machine learning, extractive summarization usually involves weighing the essential sections of sentences and using the results to generate summaries.

Several types of algorithms and methods can be used to gauge the weights of the sentences and then rank them according to their relevance and similarity with one another—and further joining them to generate a summary. Here is an example:



Source Text: Peter and Elizabeth took a taxi to attend the night party in the city.

While in the party, Elizabeth collapsed and was rushed to the hospital.

Summary: Peter

- **Abstractive Summarization:** In abstraction-based summarization, advanced deep learning techniques are applied to paraphrase and shorten the original document, just like humans do. Think of it as a pen—which produces novel sentences that may not be part of the source document.



Pen = Abstraction-based summarization

Abstractive machine learning algorithms can create new sentences that summarize the key information from the original text. This can help overcome errors in grammar that may arise when using extraction techniques. Here is an example:



Although abstraction performs better at text summarization, developing its algorithms requires complicated deep learning techniques and sophisticated language modeling. To generate plausible outputs, abstraction-based summarization approaches must address a wide variety of NLP problems, such as natural language generation, semantic representation, and inference permutation.

Text summarization is the process of creating a concise summary of text. Our project is taking video data as source input, convert to audio file, extract the text, and then extract the summary.

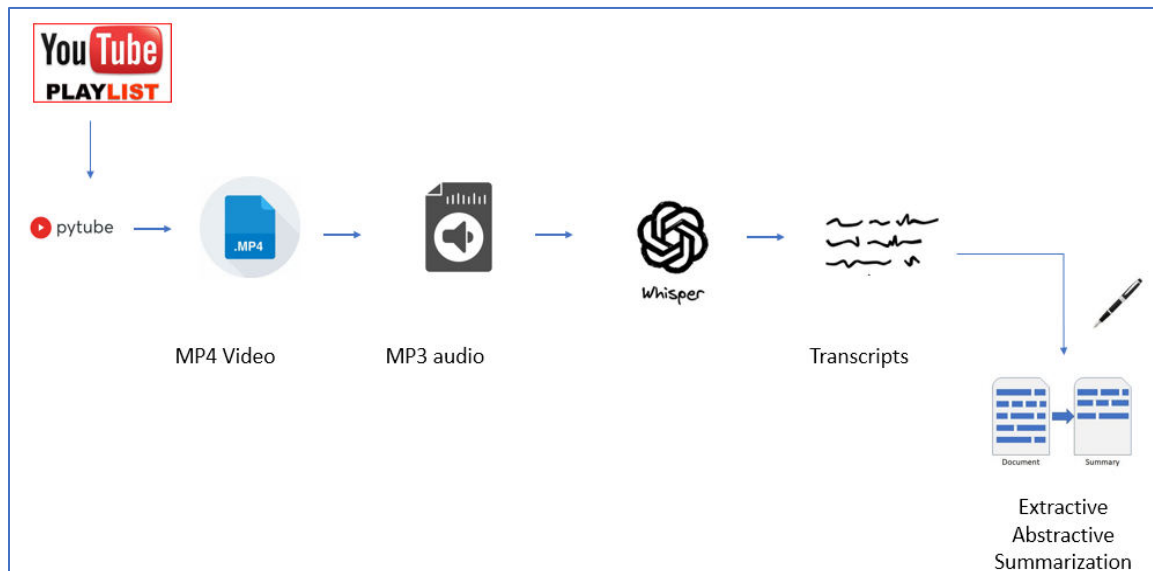


Figure 1: High-Level Architecture of the Solution

## 5. ARCHITECTURE

The architecture used is a data pipeline that processes YouTube videos to extract insights using machine learning, data transformation, and data visualization techniques. Here's a breakdown of the different components:

- PyTube is used to extract the video playlist from YouTube, which is a Python library that provides a simple and easy-to-use interface for interacting with the YouTube API.
- Whisper AI, which is now known as OpenAI, is then used to convert the videos to text transcripts.
- The transcripts are stored in an AWS S3 bucket, which is a scalable and reliable storage service provided by Amazon Web Services.
- Next, AWS Glue Data Brew is used to clean and transform the data. Data Brew is a visual data preparation tool that simplifies the process of cleaning and transforming data.
- The transformed data is sent back to the S3 bucket.
- Boto3 is used to integrate AWS with Google Colab, a cloud-based data analysis and machine learning platform. Boto3 is a Python library that provides a simple and easy-to-use interface for interacting with AWS services.
- Python code is then run on the Colab platform to carry out data modelling and analysis.
- Tableau is used for data visualization, which is a data visualization tool that allows users to create interactive and visually appealing charts and graphs.
- A video to text summary is generated using the transcript data, which provides a concise summary of the video content.
- Finally, an AWS EC2 instance is used to deploy the model and serve the video to text summary to users.

This architecture demonstrates a well-planned and organized data pipeline that involves various components, each playing a critical role in processing the data and extracting insights. By combining machine learning, data cleaning, data transformation, and data visualization techniques, this architecture allows for a thorough and in-depth analysis of the YouTube videos, which can be used to generate valuable insights for a wide range of applications.

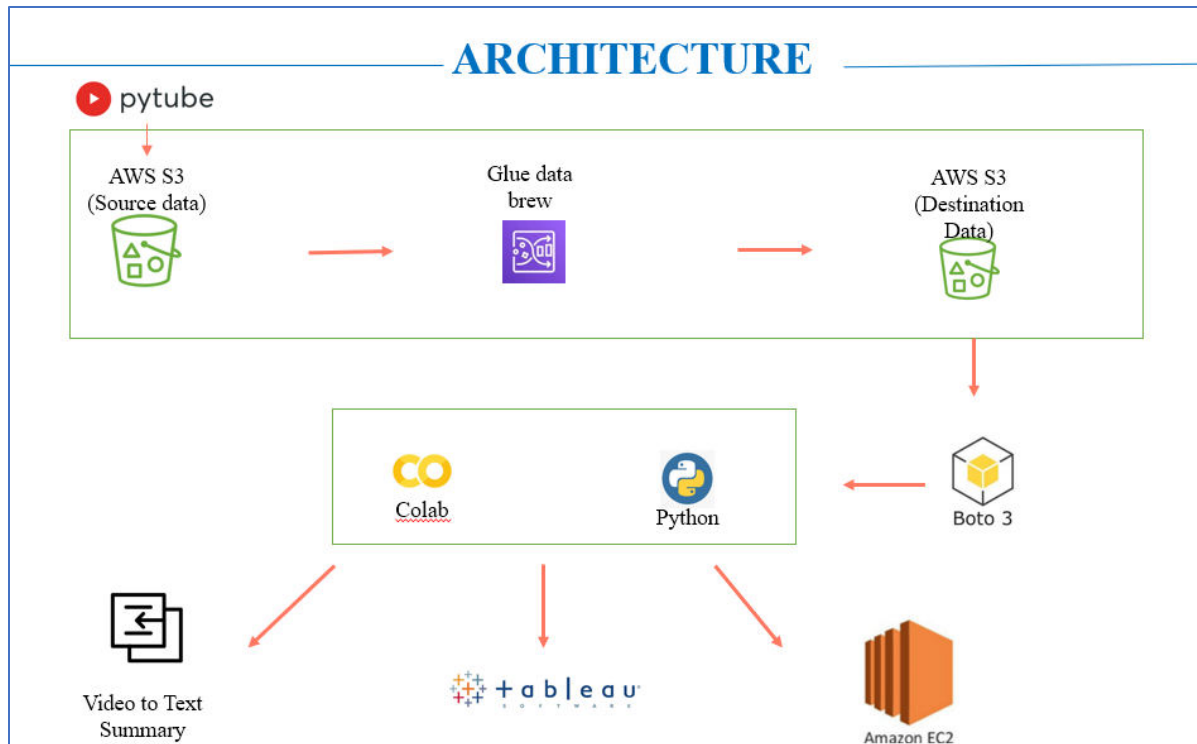


Figure 2: Architecture

## 6. DATA SOURCE

A playlist resource on YouTube represents a collection of videos that play in a specific order. These playlists usually consist of videos that are similar or related to a particular topic or subject. For our project, we are specifically utilizing playlists from educational channels on YouTube. To download these playlists, we are using the PyTube library.

### PyTube:

PyTube is a Python library that makes it easy to download YouTube videos. With PyTube, you can extract the video URL, get information about the video, and download the video to our local machines. The advantage of using PyTube are:

- **Easy to use:** PyTube has a simple and straightforward API (Application Programming Interface), making it easy to get started and download videos quickly.



- **Access to video metadata:** PyTube provides easy access to various metadata about the video, such as the title, description, and thumbnail URL.
- **Support for video streams:** PyTube provides access to different video and audio streams, making it easy to download videos in different resolutions and formats.
- **Support for video captions:** PyTube allows you to extract closed captions (subtitles) for the video if they are available.

**To obtain the YouTube videos, here are the steps we used to extract dataset:**

#### **STEP 1: Installing PyTube**

```
pip install pytube
```

#### **STEP 2: YouTube playlist Access**

First an instance of the YouTube class is created, passing in the Playlist URL of the video to be downloaded.

```
from pytube import Playlist, YouTube  
playlist_url = f'https://www.youtube.com/watch?v=Q33KBiDriJY&list=PL9ooVrP1hQO  
HPt-qWXdqElNXrNsOw_yWd'  
pl = Playlist(playlist_url)
```

#### **STEP 3: Downloading Videos**

Next, you can use the streams attribute of the YouTube class to access the different video and audio streams available for the video. The streams attribute returns a list of available streams, and you can use the `get_by_itag()` method to get any other stream format. Finally, the download method on the video object is used to download the video to the local machine.

```

for video in pl.videos:
    try:
        yt = YouTube(video.watch_url)
        video_length = yt.length//60
        if video_length < 30:
            stream = yt.streams.get_by_itag(22)
            print(f'Downloading {yt.title}...')
            stream.download(output_path='/content/videos')
            count += 1
        else:
            print(f'Skipping {yt.title} because it is too long ({video_length} minutes)...')
    except:
        print(f'Skipping {video.watch_url}...')

```

#### STEP 4: Video Metadata

In addition to downloading videos, pytube also provides easy access to various metadata about the video. The video title, description, thumbnail URL, and other information about the video can be accessed using the following code. In our project we are just accessing the title of the video.

```

yt = YouTube(video.watch.url)
print ("Video Title:", yt.title)

```

### Whisper AI

Whisper is an automatic speech recognition model trained on 680,000 hours of multilingual data collected from the web. As per Open AI, this model is robust to accents, background noise and technical language. In addition, it supports 99 different languages' transcription and translation from those languages into English.

#### STEP 1: Install whisper

This line installs the whisper library, which is used to perform audio transcription using OpenAI's models.

```

pip install git+https://github.com/openai/whisper.git

```

## STEP 2: Import

These lines import the whisper library and load the pre-trained medium model for audio transcription.

```
import whisper model = whisper.load_model("medium")
```

## STEP 3: Install ffmpeg

This line installs ffmpeg, which is a software library used for handling video, audio, and

```
!apt-get install ffmpeg
```

other multimedia files.

## STEP 4: Google Drive Mount

These lines mount the Google Drive account to the Colab notebook, allowing access to the files stored in the Drive.

```
from google.colab import drive drive.mount('/content/drive')
```

## STEP 5: Import OS

These lines import the os library and set the path variable to the directory where the audio files are stored.

```
import os path = '/content/drive/MyDrive/ProjectAudios'
```

## STEP 6: MP3 Transcription

These lines iterate through all the files in the specified directory and check if the file has a .mp3 extension. If it does, the full file path is created, and the audio file is transcribed using the loaded whisper model. The resulting transcription is printed to the console, along with the name of the file.

```

for file1 in os.listdir(path):
    if file1.endswith('.mp3'):
        # create the full file path
        file_path = os.path.join(path, file1)

        # transcribe the mp3 file
        result = model.transcribe(file_path)

        # print the transcribed text
        print("Transcription for", file1, ":", result["text"])

```

The dataset includes the following features:

Column	Column Description
Channel	This field indicates the selected channel for our project from YouTube.
MP3 File	An MP3 file is a digital audio file format used to store compressed audio data.
Transcript	A transcript is a written or typed record of spoken language.
Reference Summary	A brief and condensed summary of transcript

	Channel	MP3File	Transcripts	Reference Summary
0	TEDEd	How does heart transplant surgery work - Roni ...	What is a heart? Your heart beats more than 1...	In just a minute, it pumps over 5 liters of bl...
1	TEDEd	The myth of Hades and Persephone - Iseult Gill...	Every year before the ancient Greeks sowed the...	Every year before the ancient Greeks sowed the...
2	TEDEd	The tragic myth of the Sun Gods son - Iseult G...	Every morning Helios harnessed his winged hors...	Every morning Helios harnessed his winged hors...
3	TEDEd	What is the rarest color in nature - Victoria ...	Every color you see in front of you can be fou...	Some plant, animal, or mineral bears almost ev...
4	TEDEd	Why a sausage can do what your gloves cannot ~...	In 2010, South Korea experienced a particularl...	People couldn't activate their smartphones whi...

## DATASET SIZE:

999 records are extracted from the YouTube website.

## 7. EXTRACT, TRANSFORM AND LOAD (ETL)

We will perform ETL (Extract, Transform, Load) on the videos, and the length of video size is limited to less than 30 minutes. The first step in the ETL process involves retrieving data from YouTube Playlist channels.

**Extract:**

The initial stage of ETL is extraction, which involves obtaining data from its source before it can be moved to its destination. During this phase, unstructured data is gathered and consolidated into a single repository.

For this project, we extracted the video data using the PyTube library and converted the videos to text using Whisper AI. The real-time data was downloaded in CSV format and stored in an AWS S3 bucket. We have collected around 1200 videos and converted them to text.

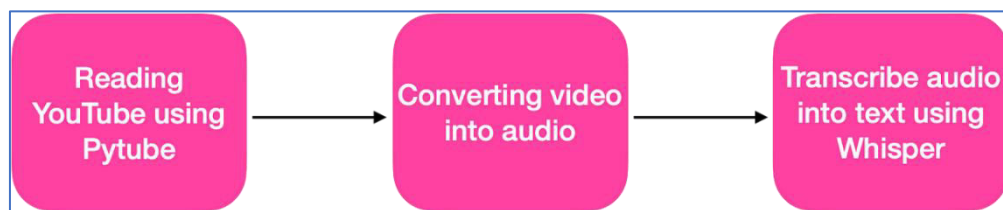


Figure 3: Techniques used for data extraction.

**Transform:**

Transformation is the process of converting the extracted data from its previous form into a form that is required to be placed in another database. Data transformations are typically accomplished through rules, lookup tables, or by combining data with other data. In this case, only one database is needed to store the original YouTube videos. However, video conversion and cleanup are performed on the local machine or directly with the actual data stored in Google Drive.

In the ETL process, it is important to apply rules and regulations during this phase to ensure data quality and accessibility. This step is crucial for improving data integrity. The transformation process involves multiple sub-processes, including cleansing, standardization, deduplication, verification, and sorting.

For this project, we used AWS Glue DataBrew to clean and transform the source data. This tool helps to automate the data preparation process, making it easier to apply rules and regulations and ensure that the data is of high quality.

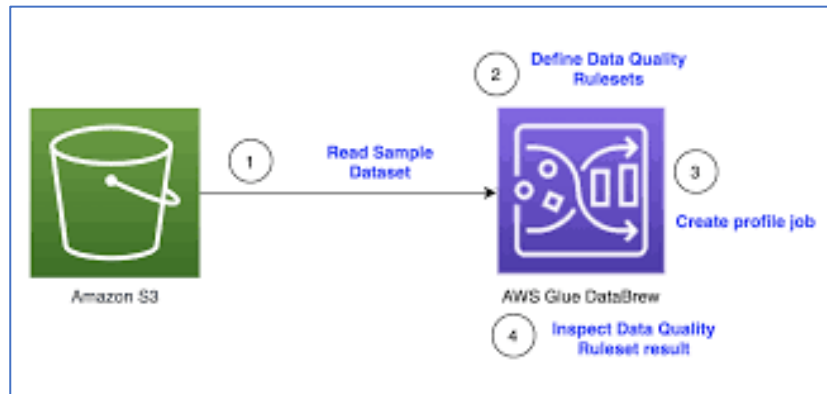
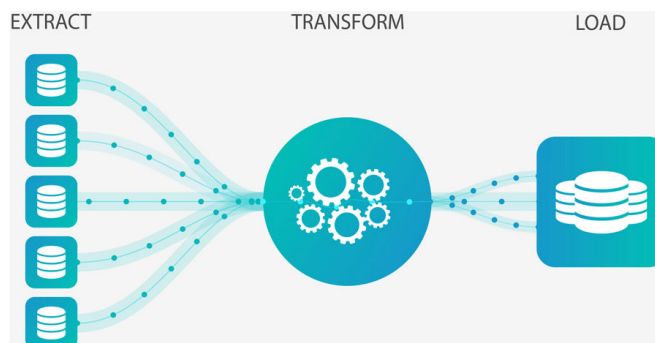


Figure 4: AWS Glue DataBrew

### Load:

The last step of the ETL process is to load the transformed data into a destination, such as a data lake or data warehouse. In this project, we loaded the processed data into an AWS S3 bucket, which serves as our data lake.



## 8. DATA CLEANING

AWS Glue DataBrew is a tool that can be utilized for both cleaning and transforming data. It provides diverse options, such as filtering, handling missing values, and removing duplicate rows. Moreover, AWS Glue DataBrew allows users to write quality rules to ensure data accuracy and consistency.

In addition to AWS Glue DataBrew, we also use ffmpeg to extract the audio track and apply the afftdn filter. This helps to remove background noise and improve the overall quality of the audio data.

**a. Clean:**

The data preprocessing step is crucial and involves various sub-steps to ensure the quality of the dataset. Since the source data is scraped from YouTube, it may contain special characters and punctuation that can be removed using the "clean" option in AWS Glue DataBrew.

**b. Conformity:**

The dataset's conformity is checked by verifying whether each column attribute's datatype is consistent using an ETL job rule.

**c. Integrity:**

Data integrity, which is crucial for maintaining data reliability and trustworthiness throughout its lifecycle, is checked using the filter option in AWS Glue DataBrew.

**d. Completeness:**

The completeness of the dataset is evaluated by checking the missing values.

**e. Accuracy:**

The accuracy percentage of the dataset is obtained by using AWS Athena and AWS QuickSight.

**f. Consistency:**

Data consistency is maintained by ensuring that the source data and processed data are consistent and in CSV format throughout the transformation process.

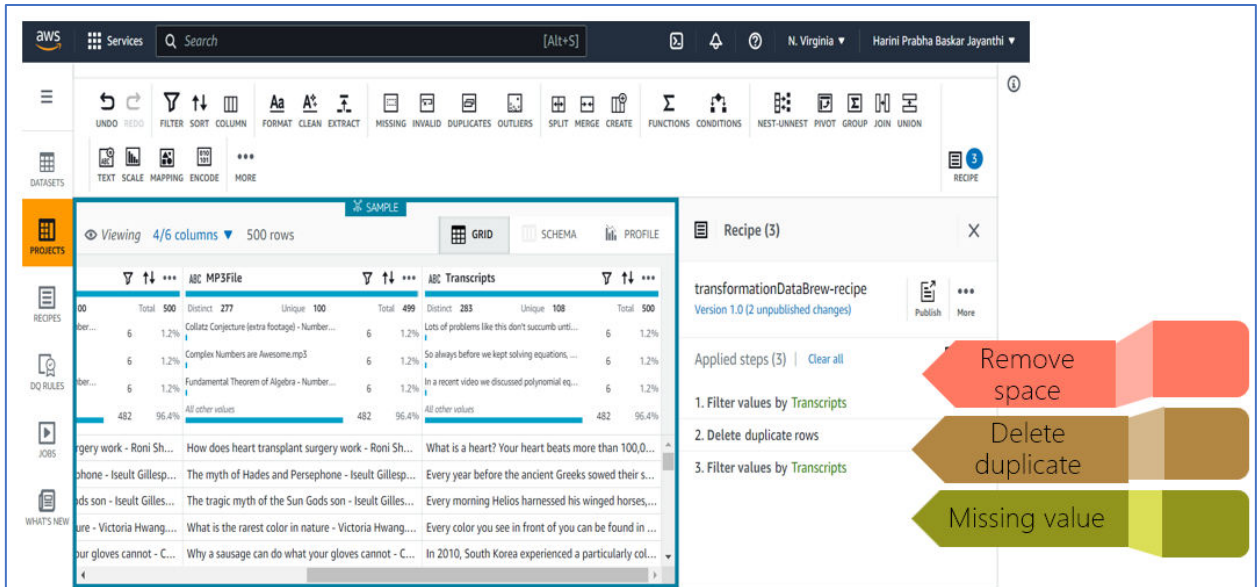


Figure 5: AWS Glue DataBrew Interface

## 9. DATA STORAGE

The extracted data is stored in an AWS Simple Storage Service (S3) bucket using a globally unique bucket name. S3 is a user-friendly object storage service that offers unmatched durability, scalability, and availability, making it an ideal location to build a Data Lake. Boto 3 is the AWS SDK for Python, which allows us to easily connect to AWS services from Colab. By using Boto 3, we can access S3 Buckets and other AWS services directly from Colab notebooks.

This allows for a streamlined workflow that enables us to quickly analyze and manipulate data in a familiar environment.

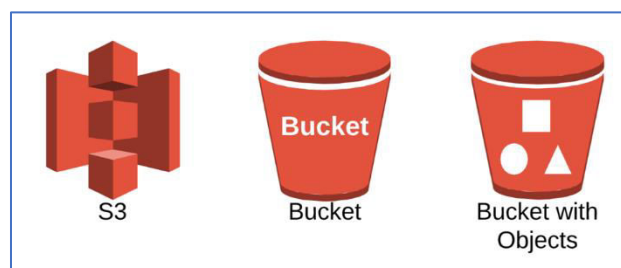


Figure 6: Amazon S3



**Following are the steps to create AWS S3 Bucket to store data:**

### **Step 1: Navigate to S3**

In the AWS console, search for S3 under Services.

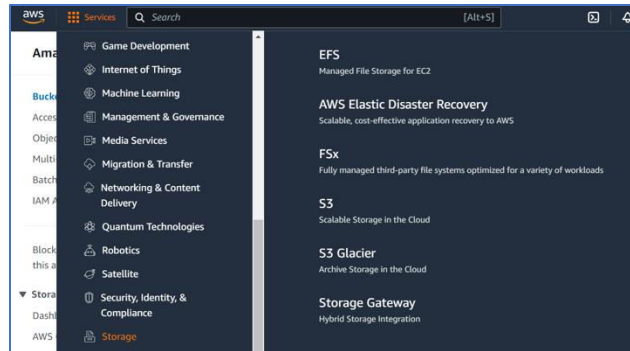
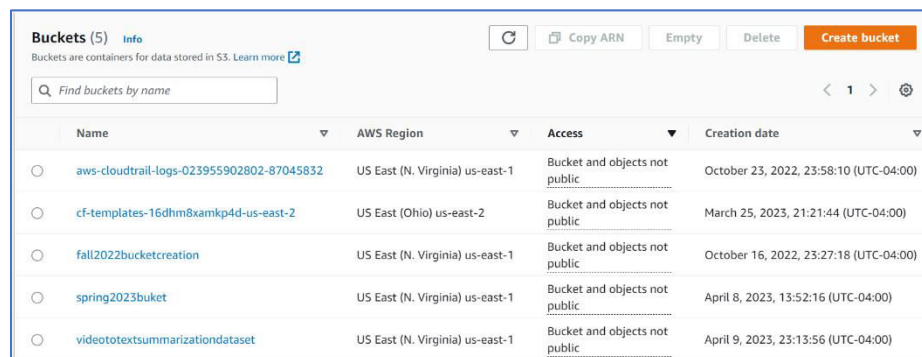


Figure 7: AWS console

### **Step 2: Create an S3 bucket**

By clicking on the “Create Bucket.” Choose a name for the bucket which should be unique, and the name cannot have spaces or uppercase letters. And then set the configuration.



Name	AWS Region	Access	Creation date
aws-cloudtrail-logs-023955902802-87045832	US East (N. Virginia) us-east-1	Bucket and objects not public	October 23, 2022, 23:58:10 (UTC-04:00)
cf-templates-16dhm8xamkp4d-us-east-2	US East (Ohio) us-east-2	Bucket and objects not public	March 25, 2023, 21:21:44 (UTC-04:00)
fall2022bucketcreation	US East (N. Virginia) us-east-1	Bucket and objects not public	October 16, 2022, 23:27:18 (UTC-04:00)
spring2023buket	US East (N. Virginia) us-east-1	Bucket and objects not public	April 8, 2023, 13:52:16 (UTC-04:00)
videototextsummarizationdataset	US East (N. Virginia) us-east-1	Bucket and objects not public	April 9, 2023, 23:13:56 (UTC-04:00)

Figure 8: AWS S3 bucket

### **Step 3: Create an Object**

Next step is to create objects in S3 bucket. These objects are entities stored in S3. Later, we can click on “Upload” option to upload files that are necessary.

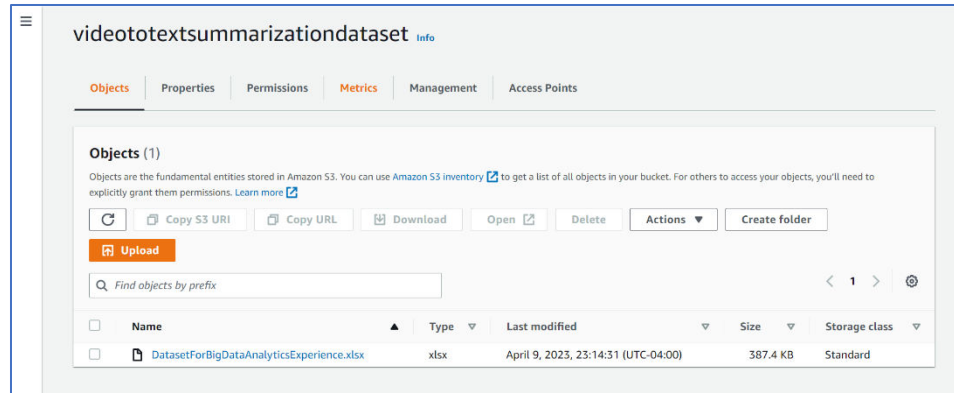


Figure 9: Create an Object interface

#### Step 4: Bucket Versioning

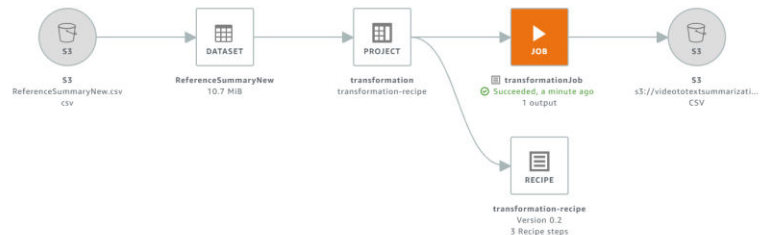
Versioning serves to protect, retrieve, and restore each version from each object recorded in an S3 bucket.



Figure 10: Bucket Versioning

## 10. DATA LINEAGE

Using AWS Data Lineage, a visual representation of the overall flow of data can be obtained. It provides a look at how the data is manipulated via the ETL process.



## 11. EXPLORATORY DATA ANALYSIS

### 1. Preprocessing Data:

The data is scraped directly from YouTube website using pytube. Once the data is scraped and stored in excel, the following steps were performed:



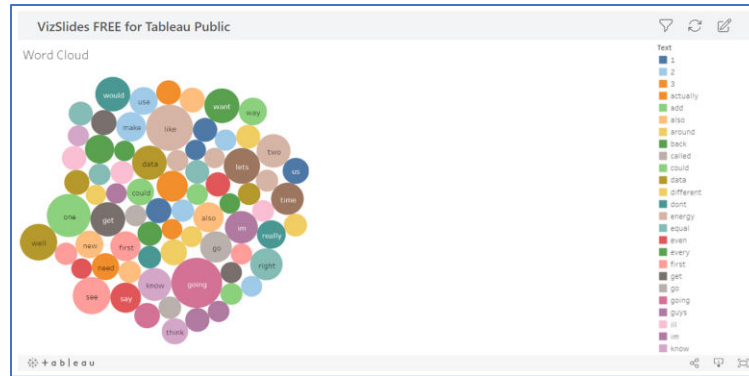


Figure 12: Tableau – Word Cloud

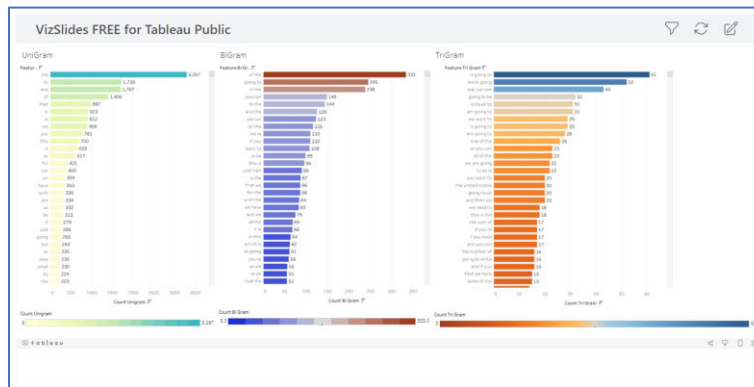


Figure 13: Tableau – UniGram, BiGram, TriGram

## 12. MODELS

We used BERTS and GPT-2 (Generative Pre trained Transformer 2) to compare the scripts.

### **BERT (Bidirectional Encoder Representations from Transformers)**

- BERT (Bidirectional Encoder Representations from Transformers) is a state-of-the-art pre-trained language model developed by Google, which uses deep learning techniques to understand natural language text. It has significantly improved the performance of various NLP tasks such as text classification, sentiment analysis, question answering, and more.
- BERT has been pre-trained on large amounts of text data, making it highly efficient and effective. It uses a transformer-based architecture, which enables it to learn the context of words in a sentence, thus making it highly effective in understanding the nuances of natural language.

- Due to its powerful pre-training, BERT has been used in various applications, including language translation, virtual assistants, and text summarization. The BERT-based summarization model used in this project was able to produce concise summaries of the input text by selecting the most relevant sentences, thus making it highly useful for applications that require quick and efficient understanding of large volumes of text data.

```
[ ] import torch
    torch.save(bert_model, '/content/drive/My Drive/Pretrained Model/bert_summarizer_modelWithReferenceSummary.pth')

[ ] # load the saved model
    saved_model_path = '/content/drive/My Drive/Pretrained Model/bert_summarizer_modelWithReferenceSummary.pth'
    bert_model = torch.load(saved_model_path)

▶ # Loop through the rows of the testing dataframe and generate summaries
  for i, row in test_df.iterrows():
      summary = bert_model(row['Transcripts'], min_length=60)
      test_df.at[i, 'SummariesBERT'] = summary
```

## GPT-2 (Generative Pre-trained Transformer 2)

- GPT-2 (Generative Pre-trained Transformer 2) is a powerful deep learning model developed by Open AI that can generate high-quality text with coherent and diverse outputs in response to a given input. It uses a transformer-based architecture and has been trained on massive amounts of text data to learn the statistical relationships between words and their context.
- Due to its ability to generate human-like text, GPT-2 has been used in various applications, such as chatbots, text completion, and even creative writing. Its high accuracy and flexibility make it a popular choice for natural language processing tasks.
- In this project, the GPT-2 model was used for text summarization, where it was able to produce concise and accurate summaries of the input text by selecting the most relevant sentences. The model's capability to generate human-like text made it highly useful in understanding the nuances of natural language, making it an ideal choice for applications that require quick and efficient processing of large volumes of text data.

```
[ ] import torch
    torch.save(GPT2_model, '/content/drive/My Drive/Pretrained Model/gpt_summarizer_modelReferenceSummary.pth')

[ ] # load the saved model
    saved_model_path_Gpt2 = '/content/drive/My Drive/Pretrained Model/gpt_summarizer_modelReferenceSummary.pth'
    gpt2_model = torch.load(saved_model_path_Gpt2)

[ ] # Loop through the rows of the testing dataframe and generate summaries
    for i, row in test_df.iterrows():
        summaryGPT2 = gpt2_model(row['Transcripts'], min_length=60)
        test_df.at[i, 'SummariesGPT2'] = summaryGPT2
```

## XLNet – Reference Summary:

XLNet (eXtreme Language understanding Network) is a pre-trained language model that is designed to understand the meaning and context of natural language text. It can perform various natural language processing tasks, such as generating relevant text for text summarization.

In the context of text summarization, it is essential to produce a reference summary that accurately captures the most important information from the source text. However, relying solely on human-generated summaries for this task is both impractical and prone to errors. XLNet can help alleviate these challenges by generating summaries that are both accurate and concise.

By pre-training on a large amount of text data, XLNet has developed a deep understanding of the statistical relationships between words and their context. This allows it to generate summaries that capture the key ideas and concepts from the source text, making it an ideal tool for various applications such as news article summarization, chatbots, and virtual assistants.

### Evaluation Metric: (ROUGE):

- ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a set of metrics used to evaluate the quality of automatic summaries generated by text summarization systems.
- There are several different versions of the ROUGE metric, each with a different focus on recall, precision, or F1 score. Some common versions of ROUGE include ROUGE-1 (unigram overlap), ROUGE-2 (bigram overlap) and ROUGE-L (longest common subsequence).
- ROUGE scores are typically reported as a set of scores for each version of the ROUGE metric, ranging from 0 to 1, with higher scores indicating better performance. The scores are computed by comparing the generated summary of BERT or GPT2 between XLNet reference summary and calculating the overlap between the generated and reference summaries based on the selected ROUGE metric.

### Bert Rouge Evaluation:

To evaluate the performance of the BERT-based summarization model, we used the

```
from rouge import Rouge

# Initialize the ROUGE metric
rouge = Rouge()

# Calculate the ROUGE scores for each row in the test set
for i, row in df.iterrows():
    reference_summary = row['Reference Summary']
    generated_summary = row['SummariesBERT']
    scores = rouge.get_scores(generated_summary, reference_summary)[0]
    df.at[i, 'ROUGE-1'] = scores['rouge-1']['f']
    df.at[i, 'ROUGE-2'] = scores['rouge-2']['f']
    df.at[i, 'ROUGE-L'] = scores['rouge-1']['f']
```

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) metric, which is commonly used for evaluating text summarization systems. Specifically, we used the ROUGE-1, ROUGE-2, and ROUGE-L measures to assess the quality of the generated summaries.

To calculate the ROUGE scores, we compared each generated summary to its corresponding reference summary in the test set. We used the Python package 'rouge' to compute the ROUGE scores for each row in the test set. The calculated ROUGE scores were then added to the DataFrame for further analysis.

### GPT-2 Rouge Evaluation:

The ROUGE metric was used to evaluate the performance of the GPT-2 model on the summarization task. The metric compares the generated summary against the reference summary and provides scores for precision, recall, and F1 score.

To calculate the ROUGE scores, we used the Python library Rouge. For each row in the test set, we obtained the reference summary and generated summary from the DataFrame. We then calculated the ROUGE scores for each summary using the Rouge library and saved the scores for each row in the DataFrame. The ROUGE scores were saved separately for unigrams (ROUGE-1), bigrams (ROUGE-2), and the longest common subsequence (ROUGE-L). These scores were used to evaluate the performance of the GPT-2 model against the reference summaries in the project.

```
from rouge import Rouge

# Initialize the ROUGE metric
rouge = Rouge()

# Calculate the ROUGE scores for each row in the test set
for i, row in df.iterrows():
    reference_summary = row['Reference Summary']
    generated_summary = row['SummariesGPT2']
    scores = rouge.get_scores(generated_summary, reference_summary)[0]
    df.at[i, 'ROUGE-1GPT2'] = scores['rouge-1']['f']
    df.at[i, 'ROUGE-2GPT2'] = scores['rouge-2']['f']
    df.at[i, 'ROUGE-LGPT2'] = scores['rouge-l']['f']
```

## 13. VISUALIZATION

The bar chart on ROUGE-1 score between generated summary of BERT and reference summary of XLNet compares the quality of the generated summaries produced by two different models, BERT and XLNet. ROUGE-1 score is a metric used to evaluate the quality of a summary by comparing it to a reference summary. Sequences the overlap of unigrams (sequences of n words) between the generated summary and the reference summary, with higher scores indicating better quality.



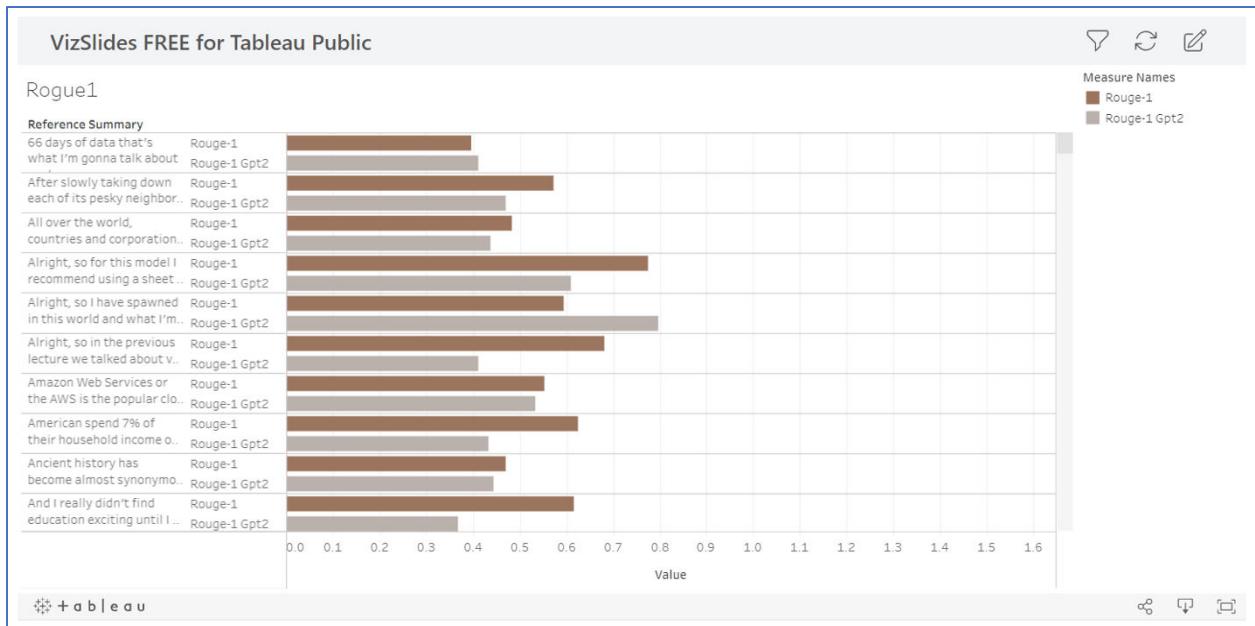


Figure 14: Tableau – bar chart on Rouge1 Score

The bar chart on ROUGE-2 score between generated summary of GPT and reference summary of XLNet compares the quality of the generated summaries produced by two different models, GPT and XLNet. ROUGE-2 score is a metric used to evaluate the quality of a summary by comparing it to a reference summary. It measures the overlap of bigrams (sequences of two words) between the generated summary and the reference summary, with higher scores indicating better quality.

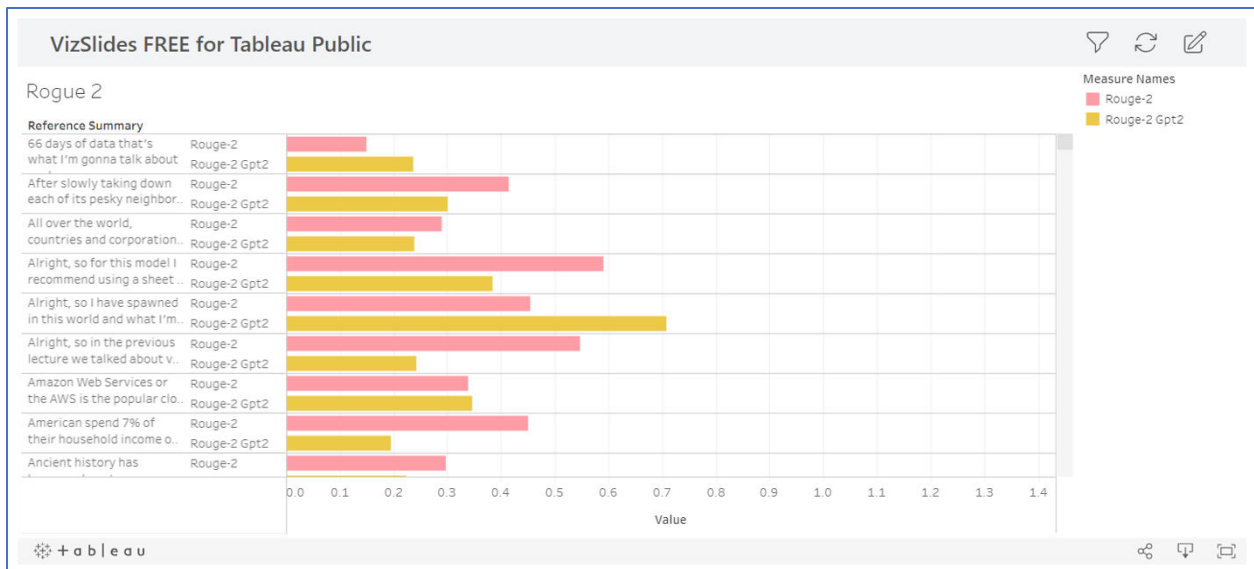


Figure 15: Tableau – bar chart on Rouge2 Score

The ROUGE-L score measures the similarity between the generated summary and the reference summary based on their longest common subsequence. Therefore, a bar chart on ROUGE-L score between generated summary of GPT and reference summary of XLNet would compare the quality of the generated summaries produced by the two different models.

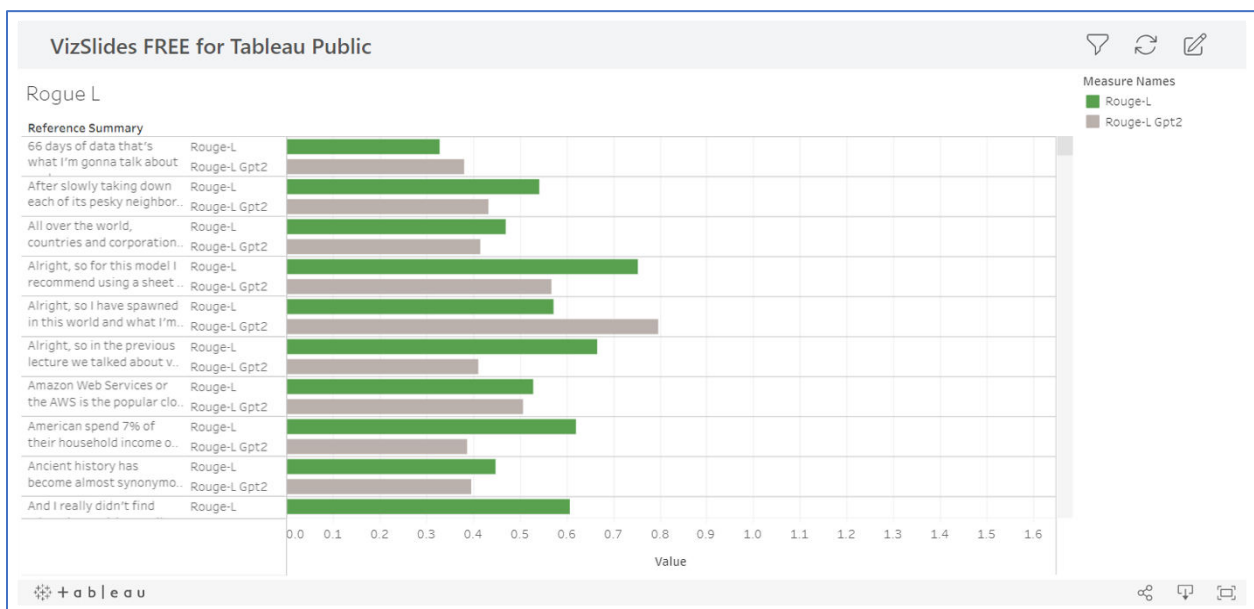


Figure 16: Tableau – bar chart on Rouge-L Score

A bar chart comparing the ROUGE-1, ROUGE-2, and ROUGE-L scores between the generated summary of BERT and reference summary of XLNet would help us compare the performance of the two models across multiple evaluation metrics.

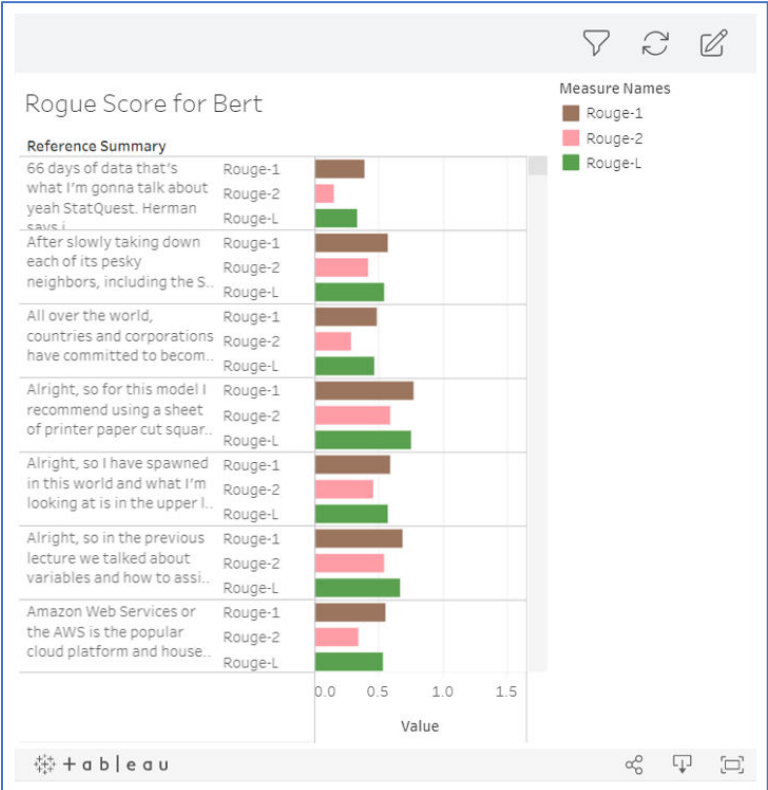


Figure 17: Tableau – bar chart on Rouge Score for Bert

A bar chart comparing the ROUGE-1, ROUGE-2, and ROUGE-L scores between the generated summary of GPT2 and reference summary of XLNet would help us compare the performance of the two models across multiple evaluation metrics.

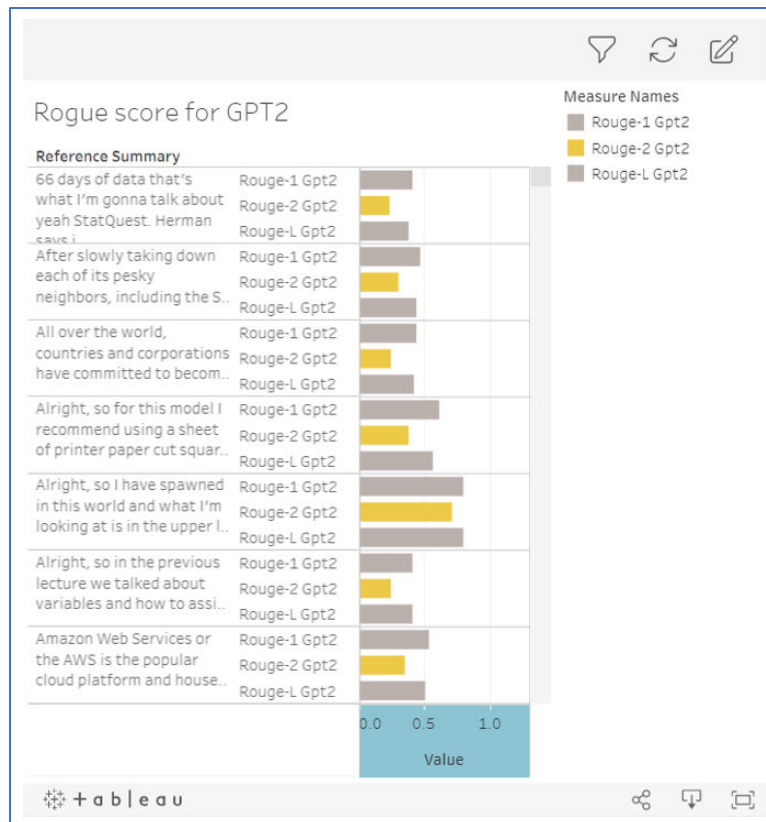


Figure 18: Tableau – bar chart on Rouge Score for GPT2(Generative Pre trained Transformer2)

A word cloud is a visualization technique that displays a collection of words in a visual representation. In a word cloud, the size and color of the words are used to represent the frequency or importance of each word in a reference summary. The most frequently occurring words are typically displayed in larger font sizes, and less frequently occurring words are displayed in smaller font sizes.

#### 14. CONNECTING GOOGLE COLAB TO AN EC2 INSTANCE:

Google Colab is a free Jupyter-based notebook environment that runs on google cloud's servers, which allows users to leverage hardware (CPU, GPU and TPU) provided by Google. Another benefit of using Colab is that it provides access to Google Drive, which can be mounted and accessed via the file explorer.

Google Colab includes a 12GB NVIDIA Tesla K80 GPU with a Nvidia compute capability of 3.7 that can be used continuously for up to 12 hours which is great for experimenting and prototyping but may become limiting for working with larger datasets and/or larger networks.

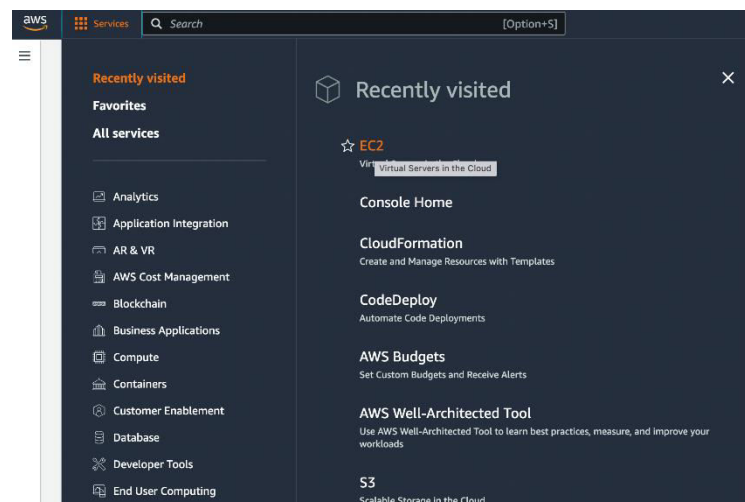
The idea is to integrate Google Colab with AWS EC2 instance so that code can run on top of the EC2 instance runtime. Connecting an external runtime to Colab simply allows one to keep the Colab interface while using GPU acceleration other than what Colab provides by default.

### Steps for Integration:

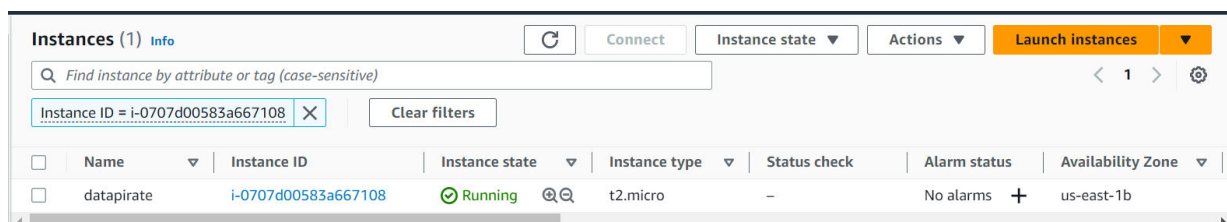
The integration of Google Colab and AWS EC2 instance is based on the following steps:

**Step 1:** Log into the AWS management console.

**Step 2:** Click on Services on the top left corner and select EC2.



**Step 3:** Select Instances in the left pane, and then Launch Instance.



**Step 4:** Choose the image type. This determines the OS and software versions that'll come preinstalled. Here we choose an Ubuntu 20.04 installation with PyTorch 2.0.0.

▼ Instance details <a href="#">Info</a>		
Platform Ubuntu (Inferred)	AMI ID ami-0a4caa099fc23090f	Monitoring disabled
Platform details Linux/UNIX	AMI name Deep Learning AMI GPU PyTorch 2.0.0 (Ubuntu 20.04) 20230401	Termination protection Disabled
Stop protection Disabled	Launch time Sun Apr 30 2023 16:18:33 GMT-0400 (Eastern Daylight Time) (2 minutes)	AMI location amazon/Deep Learning AMI GPU PyTorch 2.0.0 (Ubuntu 20.04) 20230401

## Connect via SSH

Click on the SSH client tab to see the ssh login details for the instance.

### Connect to instance [Info](#)

Connect to your instance i-0707d00583a667108 (datapirate) using any of these options

[EC2 Instance Connect](#) [Session Manager](#) [SSH client](#) [EC2 serial console](#)

Instance ID  
i-0707d00583a667108 (datapirate)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is datapiratekeypair.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.  
chmod 400 datapiratekeypair.pem
4. Connect to your instance using its Public DNS:  
ec2-3-84-236-99.compute-1.amazonaws.com

Example:  
ssh -i "datapiratekeypair.pem" ubuntu@ec2-3-84-236-99.compute-1.amazonaws.com

**Note:** In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

## SSH Details

Putty SSH client has been used to login to the instance. The main reason using Putty is to set up SSH Tunneling. This is required to connect Colab to the EC2 runtime.

```
ubuntu@ip-172-31-90-250: ~
Using username "ubuntu".
Authenticating with public key "imported-openssh-key"

      _ _      _
     _(_)_    /
    _\_\_\_\_/_

Deep Learning AMI GPU PyTorch 2.0.0 (Ubuntu 20.04)

=====
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-1033-aws x86_64v)

* Please note that Amazon EC2 P2 Instance is not supported on current DLAMI.
* Supported EC2 instances: G3, P3, P3dn, P4d, P4de, G5, G4dn.
* To activate pre-built pytorch environment, run: 'source activate pytorch'
* To activate base conda environment upon login, run: 'conda config --set auto_activate_base true'
* NVIDIA driver version: 525.85.12
* CUDA version: 11.8

AWS Deep Learning AMI Homepage: https://aws.amazon.com/machine-learning/amis/
Release Notes: https://docs.aws.amazon.com/dlami/latest/devguide/appendix-ami-release-notes.html
Support: https://forums.aws.amazon.com/forum.jspa?forumID=263
For a fully managed experience, check out Amazon SageMaker at https://aws.amazon.com/sagemaker
```

The jupyter\_http\_over\_ws extension authored by Google Colab needs to be installed next to allow connection to Colab.

```
pip install jupyter_http_over_ws
jupyter serverextension enable --py jupyter_http_over_ws
```

Run the commands below in the command line:

Use the command below to start the Jupyter Notebook service on the EC2 instance:

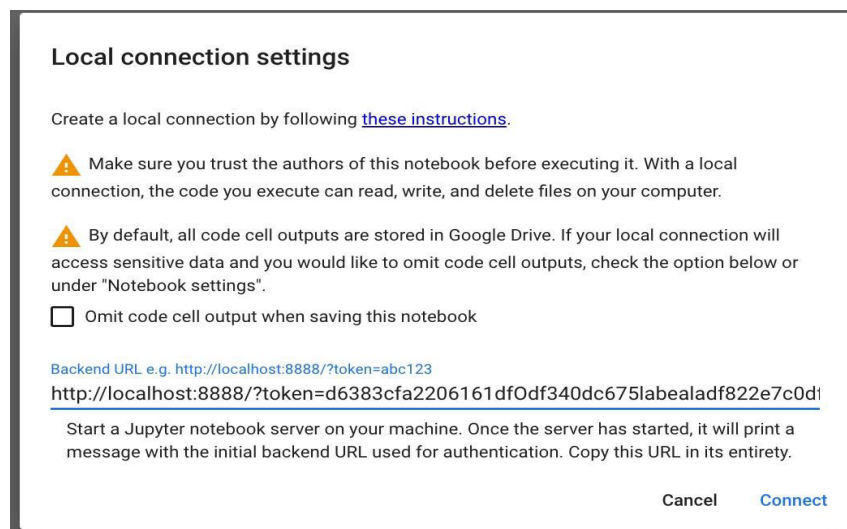
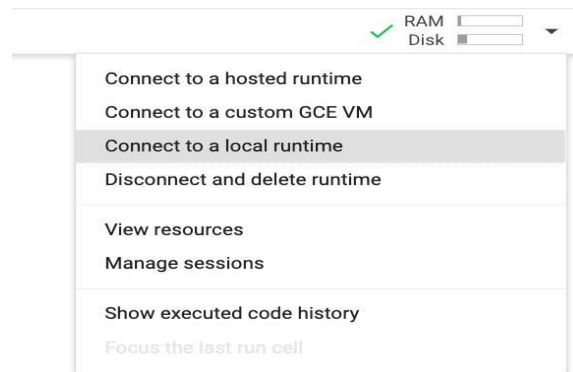
```
jupyter notebook \
--NotebookApp.allow_origin='https://colab.research.google.com' \
--port=8888 \
--NotebookApp.port_retries=0
```

```
[I 16:40:13.803 NotebookApp] Jupyter Notebook 6.5.3 is running at:
[I 16:40:13.803 NotebookApp] http://localhost:8888/?token=bdf5ec5c17b67ad79733bb8c31b2b93ba41a75b232730218
[I 16:40:13.803 NotebookApp] or http://127.0.0.1:8888/?token=bdf5ec5c17b67ad79733bb8c31b2b93ba41a75b232730218
[I 16:40:13.803 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[W 16:40:13.807 NotebookApp] No web browser found: could not locate runnable browser.
[C 16:40:13.807 NotebookApp]

To access the notebook, open this file in a browser:
file:///home/ubuntu/.local/share/jupyter/runtime/nbserver-26326-open.html
Or copy and paste one of these URLs:
http://localhost:8888/?token=bdf5ec5c17b67ad79733bb8c31b2b93ba41a75b232730218
or http://127.0.0.1:8888/?token=bdf5ec5c17b67ad79733bb8c31b2b93ba41a75b232730218
^C[I 16:43:22.425 NotebookApp] interrupted
Serving notebooks from local directory: /home/ubuntu
0 active kernels
Jupyter Notebook 6.5.3 is running at:
http://localhost:8888/?token=bdf5ec5c17b67ad79733bb8c31b2b93ba41a75b232730218
or http://127.0.0.1:8888/?token=bdf5ec5c17b67ad79733bb8c31b2b93ba41a75b232730218
Shutdown this notebook server (y/[n])? n
resuming operation...
```

If all goes well, an IP address will be obtained. The address should be copied as it will be needed in the next step.

Open Google Colab, in the top right-hand corner, select Connect to local runtime.



Copy the IP address from the terminal and click Connect. Finally, Colab is connected to a runtime other than the Colab default.

## 15. CHALLENGES

- **Downloading multiple videos from different channels:** Collecting video content from various sources can be challenging, especially when dealing with a large amount of video data.



- **Evaluating the quality of the summary is difficult:** Unlike text-based summaries, evaluating the quality of video-based summaries can be subjective and challenging.
- **Videos can be diverse:** Videos can have different genres, styles, and lengths, making it challenging to build a summarization model that works well for all types of videos.
- **Extracting relevant information:** Extracting relevant information from videos can be challenging, especially when there are multiple speakers or when the video contains background noise.
- **Generating coherent and informative summaries:** Generating a coherent and informative summary that captures the essential points of the video while preserving its context and flow can be challenging.
- **Handling non-verbal cues:** Videos often contain non-verbal cues such as facial expressions, body language, and tone of voice, which can be essential in understanding the content but can be difficult to capture and summarize using text.

## 16. INDIVIDUAL CONCLUSION AND LEARNING:

- This project involves the use of different technologies to extract insights from YouTube videos. The pipeline consists of several components that work together to achieve this goal.
- The first step involves using PyTube to extract the video playlist from YouTube.
- Ffmpeg is then used to convert the video to audio and enhance the audio by reducing noise using the `afstdn` parameter.
- The next step is to use Whisper AI to convert the videos to text transcripts, which are then stored in an AWS S3 bucket.
- The data in the S3 bucket is then cleaned and transformed using AWS Glue DataBrew by creating recipes and creating and running jobs. This is a visual data preparation tool that simplifies the process of cleaning and transforming the data. Since our dataset was already clean, we did basic cleaning like checking for missing values and deleting the duplicate records. The transformed data is then sent back to the S3 bucket.
- Next, Boto3 is used to integrate AWS with Google Colab, a cloud-based data analysis and machine learning platform, where Python code is run for data modelling and analysis.
- Boto3 requires authentication to access and manipulate AWS resources. It uses a set of AWS credentials to authenticate with AWS services. These credentials include an access key and a secret access key, which are used to request AWS services and prove the identity of the user.
- To provide AWS credentials to Boto3 we used IAM roles. An IAM role was created specifying the permissions and an access key was generated under the security credentials tab by access keys.

- Once Colab was integrated with AWS using Boto3 we used some functions to call the S3 bucket to Colab and perform modelling.
- Since we needed a reference summary (a human generated summary) for our evaluation, we used the XLNet model generated summary as our reference summary.
- We split our dataset to 80-20 ratio to evaluate our model.
- We used two models BERT (Bidirectional Transformer) and GPT-2 (Generative Pre-trained Transformer 2) to generate the summary.
- We evaluated our model generated summary with reference summary using ROUGE score.
- Pictures speak louder than words, so we did visualization using Tableau to understand the dataset better. We visualized ROUGE scores between two models and did visualization on word cloud and n-gram on reference summary data.
- Finally, an EC2 instance was created with Colab to overcome the limitations on the computing resources. In our project we did not run out of computing resources as we used Colab Pro but to get hands-on experience, we created an EC2 instance and connected it with Colab.
- After creating EC2 instance, we made an SSH connection with putty. The main reason we used Putty was to set up SSH Tunneling, which was necessary to get Colab to connect to the EC2 runtime. While creating an instance, we created key file in AWS which is .pem format. We loaded this file in PuttyGen to convert .pem file to .ppk file so that Putty can recognize this file format for authentication purpose.
- We also logged into the EC2 instance with putty using SSH login. After successful login, a command line was generated to control the EC2 instance. We used few commands to start the Jupyter notebook service on the EC2 instance. An IP address was generated on the terminal screen. We copied this IP address to Colab and finally, Colab was connected to a runtime other than Colab default.
- Overall, the project aims to automate the process of summarizing the content of YouTube videos in text form, using machine learning and data processing techniques.

## KEY TAKEAWAYS

- Initially we attempted to manually download videos from YouTube and then convert them to text, but we were unsuccessful due to account restrictions. However, we discovered how to automatically download more videos in one click using the PyTube library.
- I learnt some AWS services before but having hands-on experience gave me in depth knowledge about them.
- The use of IAM roles for authentication to access and manipulate AWS resources with Boto3 was a key takeaway.
- Use of SSH tunneling to connect to an EC2 instance to Colab using Putty was another important takeaway.
- As someone interested in data engineering, creating an ETL pipeline was another important takeaway for me.

## 17. REFERENCES

- <https://towardsdatascience.com/a-quick-introduction-to-text-summarization-in-machine-learning-3d27ccf18a9f>
- [https://www.itm-conferences.org/articles/itmconf/pdf/2022/04/itmconf\\_icacc2022\\_03063.pdf](https://www.itm-conferences.org/articles/itmconf/pdf/2022/04/itmconf_icacc2022_03063.pdf)
- <https://towardsdatascience.com/extractive-summarization-using-bert-966e912f4142>
- <https://medium.com/analytics-vidhya/text-summarization-using-bert-gpt2-xlnet-5ee80608e961>
- <https://towardsdatascience.com/connecting-google-colab-to-an-amazon-ec2-instance-b61be9f9cf30>