ABSTRACT

Integrated video data presentations may allow active video browsing. Such presentations provide the user with information about the content of a particular sequence being tested while maintaining an important message. We suggest how to automatically make video summaries for longer videos. Our video access method involves two tasks: first, splitting the video into smaller, compatible parts and second, setting the levels into effects. Our proposed algorithm sections are based on analysis of word frequency in speech transcripts. After that the summary is made by selecting the parts with the highest scores depending on the length of time and these are illustrated. We created and conducted a user study to check the quality of the summaries made. Comparisons are made using our proposed algorithm and a random segment selection scheme based on mathematical analysis of user learning outcomes. Finally, we can see the summarized context of the video we want to know about. Summarization of the video is done by the Python API and NLP (Natural Language Processing). An API, or Application Programming Interface, is a server you can use to receive and send data using code. APIs are widely used to retrieve data, and that will be the focus of this first study. When we want to receive data from an API, we need to make a request. Applications are used across the web.

Keywords:- Natural Language Processing, Machine Learning, Abstractive Text Summarization, Transformers.

CHAPTER-1

1. INTRODUCTION

YouTube is a video sharing platform, the second most visited website, the second most used search engine, and is stronger than ever after more than 17 years of being online. YouTube uploads about 720,000 hours of fresh video content per day. The number of videos available on the web platform is steadily growing. It has become increasing easy to watch videos on YouTube for anything, from cooking videos to dance videos to motivational videos and other bizarre stuff as well. The content is available worldwide primarily for educational purposes. The biggest challenge while extracting information from a video is that the viewer has to watch the entire video to understand the context, unlike images, where data can be gathered from a single frame. If a viewer has low network speed or any other device limitation can lead to watch video with a low resolution that makes it blurry and hectic to watch. Also, in between advertisements are too frustrating. So, removing the junk at the start and end of the concerned video as well as skipping advertisements, and getting is summary to directly jump to your part of interest is valuable and time efficient.

This project focuses on to reducing the length of the script for the videos. Summarizing transcripts of such videos automatically allows one to quickly lookout for the important patterns in the video and helps to save time and effort to go through the whole content of the video. The most important part of this project will be its ability to string together all the necessary information and concentrate it into a small paragraph. Video summarization is the process of identifying the significant segments of the video and produce output video whose content represents the entire input video. It has advantages like reducing the storage space used for the video. This project will give an opportunity to have hands-on experience with state-of-the-art NLP technique for abstractive text summarization and implement an interesting idea suitable for intermediates and a refreshing hobby project for professionals.

1.1 Objectives

In this project, you will be creating a Chrome Extension that will apply to the backend REST API where it will do NLP and respond with a summary of YouTube text.

1.2 Project Context

Summarizer is a Chrome extension that works with YouTube to extract the key points of a video and make them accessible to the user. The summary is customizable per user's request, allowing varying extents of summarization. Key points from the summarization process, together with corresponding time-stamps, are then presented to the user through a small UI next to the video feed. This allows the user to navigate to more important sections of the video, to get to the key points more efficiently.

1.3 Contributions

We first set up a GitHub repo for project management and made a readme to keep track of dependencies and environment information. Out of the numerous challenges we encountered throughout this hackathon, the most significant challenge was an overestimation of the technologies available to us. Our overestimation of the ability of transcription services forced us to make compromises, while our attempts to get the YouTube player to control playback pressed us to create a hacky solution. Overcoming these challenges and bridging the capabilities of these technologies was an integral part of the project.

We spend a noticeable amount of our weekly time watching YouTube videos, be it for entertainment, education, or exploring our interests. In most cases, the overall intent is to obtain some form of information from the video. We were seeking a solution to increase the efficiency of this "information extraction" process as YouTube's speed adjustment option is the only relevant tool. And so we decided to develop YouTube Summarizer!

1.4 Problem Statement

Throughout the day, an increasing number of video recordings are generated and shared on the Internet. It has become quite difficult to devote time to watching movies that may last longer than expected, and our efforts may be in vain if we are unable to extract useful information from them. Summarizing transcripts of such movies automatically allows us to rapidly spot essential patterns in the video, saving us time and effort.

CHAPTER-2

2. LITERATURE SURVEY

The Field of NLP is a field under artificial intelligence, which is used to classify, and process text with greater efficiency and accuracy than humans. While NLP as a technology can be a great technology for the love of many businesses, NLP APIs are an easy way for companies, organization, group or a single human to integrate technology into business and many more processes.

Integrating NLP APIs with existing business plan, software assisted companies to increase the efficiency and effectiveness of business processes. Its all about teaching machines to understand how to understand human languages and extract meaning from text. It provides developers with extensive collection of NLP tools and libraries that enables developers to handle a great number of NLP related tasks such as document classification, topic modeling, part of speech (POS) tagging, word vectors, and sentiment analysis. Modern technology can analyze more details about language than humans, without exhaustion and in a consistent, impartial way. Given the staggering amount of informal data generated on a daily basis, from medical records to social media platforms, automation will be essential to fully analyze text and speech data effectively.

Human language is amazingly complex and diverse. We express ourselves in endless ways, verbally and in writing. Not only are there thousands of languages and dialects, but within each language there is a different set of grammar and syntax rules, rules and slang. When we write, we often mispronounce words or abbreviate words, or leave punctuation marks. When we speak, we have ways of speaking in the region, and we mix, help and borrow words in other languages.

While both supervised and supervised learning, and especially in-depth reading, are now widely used in modeling the human language, there is also a need for a well crafted understanding and background technology that is not present in these electronic learning methods. In this paper, authors surveyed the video classification literature. They find that features are drawn from three modalities—text, audio, and visual—and that a large variety of combinations of features and classification have been explored. They also described the general features chosen and summarize the research in this area. We conclude with ideas for further research.

2.1 Back-End

APIs changed the way we build applications, there are countless examples of APIs in the world, and many ways to structure or set up our APIs . We install the dependencies like YouTube_transcript_API and transformers using pip.

Requirements

Install the following dependencies:

1. youtube_transcript_api
2. transformers

Expected Outcome

You are expected to initialize the back-end portion of your application with the required dependencies

2.2 GET TRANSCRPT FOR A VIDEO

We will utilize a python API which allows transcripts/subtitles for a given YouTube video. It also works for automatically generated subtitles, supports translating subtitles, and does not require a headless browser like other Selenium-based solutions do.

Requirements

- Create a function which will accept YouTube video id as input parameter and return parsed full transcript as output.
- The response from the Transcript API will return a list of dictionaries.

Expected Outcome

You should be able to fetch the transcripts with the help of a function created which we will later utilize as a feed input for the NLP processor in the pipline.

2.3 PERFORM TEXT SUMMARIZATION

Text summarization is the task of shortening long pieces of text into a concise summary that preserves key information content and overall meaning. There are two different approaches that are widely used for text summarization:

- **Extractive Summarization**: This is where the model identifies the important sentences and phrases from the original text and only output those.
- **Abstractive Summarization**: The model produces a completely different text that is shorter than the original; it generates new sentences in a new form, just like humans do. Here we'll use transformers for this approach. We will use HuggingFace's transformers library in Python to perform abstractive text summarization on the transcript previously obtained.

Requirements

- Create a function which will accept YouTube transcript as an input parameter and return summarized script as output.
- Instantiate a tokenizer and a model from the checkpoint name.
- Summarization is usually done using encoder-decoder model, such as BART or T5.
- Define the transcript that should be summarized.
- Add the T5 specific prefix "summarize:"
- Use the Pre-Trained Model.generate() method to generate the summary.

Expected Outcome

You should be able to verify that the model generates a completely new summarized text that is different from the original text.

2.4 CREATE REST API ENDPOINT

The next step is to define the resources that will be exposed by this backend service. This is an extremely simple application, we only have single endpoint, so our only resource will be summarized text.

Requirements

- Create a Flask API Route with GET HTTP Request method with a URI, http://[hostname]/api/summarize?youtube url=.
- Extract the YouTube from YouTube URL which is obtained from the query.
- Generate the summarized transcript by executing the transcript generation function following the execution of the transcript summarizer function.

- Return the summarized transcript with HTTP Status OK and handle HTTP exception if applicable.
- Run the Flask Application and test the endpoint in Postman to verify the appropriate results.

Expected Outcome

You should be able to create an endpoint to summarize YouTube video transcripts and test the response with different video URLs.

2.5 Display Summarized transcript

We have provided a basic UI to enable users to share and display a summary text but there are missing links that need to be fixed. In this archive, we will add functionality to allow the extension to connect to the recurring server using HTTP REST API calls.

Requirements

- In reverse operation, extract the URL of the current tab and apply GET HTTP using the XML HTTP Request Web API back to get a summary text in response.
- Send action message with short uploads using chrome to display a summary text.

Expected Outcome

The visual interface of the user should be able to display a summary text at the request of the user.

CHAPTER-3

3. SYSTEM ANALYSIS

3.1 Existing system

The existing system Text summarization takes care of choosing the most significant portions of text and generates coherent summaries that express the main intent of the given document. The services offered by our text summarizer is , summarizing the text from input, summarizing web articles or weblinks and summarizing from the PDF. Our system does not ask for user details. It provides a platform to get summary without creating an account.

This system involves the different module to generate the summary for given multiple documents. Previous system has some drawback such that it can take only text file as input. If we give other files such as PDF or word file as input then it cannot accept that file and shows the message only text files are allowed. To overcome these problems we proposed a new system that takes the input as text, PDF and weblinks. The system involves the following basic three phases. This model uses NLP based algorithms for solving problem. It does web scrapping for getting text from the web links. The text from the pdf is extracted using python Py2pdf library. Summarizer file is made which creates different functions for different purpose. The system model is defined below.

3.1.1 Disadvantages

- Text summarization is the possibility of missing relevant information. If the original piece of content was written by someone else, then the summary generator might omit key information the writer meant the reader to see. As a result, the generated summary may not be useful.
- Efforts put into training the models might not exactly meet the required standards: Neural Network-based models require large resources and time to train. The results might not exactly meet the required standards or the level of manual text summarization.
- Grammatical mistakes - abstractive algorithms are prone to grammatical mistakes: Abstractive methods rewrite certain portions of sentences to generate the summary.
- If the obtained information in not accurate then we loose our time.

3.2 Proposed system

The system takes the input YouTube video from the Chrome extension of the Google Chrome browser when the user clicks on the summarize button on the chrome extension web page, and access the transcripts of that video with the help of python API. The accessed transcripts are then summarized with the transformers package. Then the summarized text is shown to the user on the web page. A lot of technical and educational applications involving generation of large amounts of video and multimedia are top contender of using video summarization technique. These include film industry, advertisement creation, data visualization, match highlights of sports match thus removing redundancy, reducing computational time and storage requirements.

Video transcript summarizer has got a lot of scope in today's world. It highlights the important topics from the video. People spend a noticeable amount of time binge watching YouTube videos, be it for entertainment, education purposes, or getting some important information or exploring their interests. If you wish to find a video to get any important information about a topic, it is a very difficult task to achieve as most of the videos are filled with insignificant buffer material. In most of the cases, the overall intent is to obtain some form of quality information from the video. This project brings forward a video summarization system based on Natural Language Processing and Machine Learning to generalize YouTube video transcripts for abstractive text summarization without losing the main elements and content. This project focuses on to reducing the length of the script for the videos.

3.2.1 Advantages

- This model produces a completely different text that is shorter than the original, it generates new sentences in a new form.
- Time saving process.
- Looks at the whole content of the project.
- Abstractive summarization considers the entire text and creates a summary based on the main ideas of the text.
- This type of summarization is more accurate than extractive ones because it considers all the core ideas and will distill these into a short consumable text.

3.3 Modules

3.3.1 User Module

1. Input the valid YouTube video URL from the user.

The First step is to getting the video link from the user which user wants to summarize. The video should be Recorded, it should have a valid video id and it should be available on YouTube. These are some of the things that user should keep in mind before using this web application.

3.3.2 Admin Module

1.Getting the transcripts from that video.

After taking the video link from the user, the next part is to get the transcripts on video. Now it will check whether the given video has subtitles available or not. If the Given video has subtitles in it then we can simply use the python library called YouTube_Transcript_API which is used to extract the transcripts from the given video. Now what if the given video doesn't have subtitles in it so in this case we will first convert the video into audio format like .mp3 or .mp4 format using python library called pytube and then download that audio file. After downloading the audio file, there is a need to convert it into .wav file using ffmpeg and then doing its audio Transcription using Speech Recognition Library and Hugging sound Library. It will basically convert that audio file into text file which contains the Transcription of the video. This is how to get the Transcription of any video.

2. Passing the Generated transcripts to the text summarizer.

Now this is the main phase of the project where the whole project depends upon. This phase basically includes the text summarization. Now there are mainly two types of summarization techniques:

- ❖ Extractive Summarization: It will not produce any new sentence. Ex: Text Rank, LSA.
- ❖ Abstractive Summarization: It will basically reproduce the sentence which is more clean and concise than the original sentence and in a most human readable form. This is better than extractive summarization techniques. This can be easily implemented by using BART Transformers or pipeline api.

CHAPTER-4

4. FEASIBILITY STUDY

A feasibility study is a high-level capsule version of the entire System analysis and Design Process. The study begins by classifying the problem definition. Feasibility is to determine if it's worth doing. Once an acceptance problem definition has been generated, the analyst develops a logical model of the system. A search for alternatives is analyzed carefully. There are 3 parts in feasibility study.

1) OPERATIONAL FEASIBILITY

2) TECHNICAL FEASIBILITY

3) ECONOMICAL FEASIBILITY

4.1 OPERATIONAL FEASIBILITY

Operational feasibility is the measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development.The operational feasibility assessment focuses on the degree to which the proposed development projects fits in with the existing business environment and objectives with regard to development schedule, delivery date, corporate culture and existing business processes.To ensure success, desired operational outcomes must be imparted during design and development. These include such design-dependent parameters as reliability, maintainability, supportability, usability, producibility, disposability, sustainability, affordability and others. These parameters are required to be considered at the early stages of design if desired operational behaviour are to be realized. A system design and development requires appropriate and timely application of engineering and management efforts to meet the previously mentioned parameters. A system may serve its intended purpose most effectively when its technical and operating characteristics are engineered into the design. Therefore, operational feasibility is a critical aspect of systems engineering that needs to be an integral part of the early design phases.

3.2 TECHNICAL FEASIBILITY

This involves questions such as whether the technology needed for the system exists, how difficult it will be to build, and whether the firm has enough experience using that technology. The assessment is based on outline design of system requirements in terms of input, processes, output, fields, programs and procedures. This can be qualified in terms of volume of data, trends, frequency of updating in order to give an introduction to the technical 3 system. The application is the fact that it has been developed on windows XP platform and a high configuration of 1GB RAM on Intel Pentium Dual core processor. This is technically feasible .The technical feasibility assessment is focused on gaining an understanding of the present technical resources of the organization and their applicability to the expected needs of the proposed system. It is an evaluation of the hardware and software and how it meets the need of the proposed system.
According to Roger S. Pressman, Technical Feasibility is the assessment of the technical resources of the organization. The organization needs IBM compatible machines with a graphical web browser connected to the Internet and Intranet. The system is developed for platform Independent environment. Java Server Pages, JavaScript, HTML, SQL server and WebLogic Server are used to develop the system. The technical feasibility has been carried out. The system is technically feasible for development and can be developed with the existing facility.

3.3 ECONOMICAL FEASIBILITY

Establishing the cost-effectiveness of the proposed system i.e. if the benefits do not outweigh the costs then it is not worth going ahead. In the fast paced world today there is a great need of online social networking facilities. Thus the benefits of this project in the current scenario make it economically feasible. The purpose of the economic feasibility assessment is to determine the positive economic benefits to the organization that the proposed system will provide. It includes quantification and identification of all the benefits expected. This assessment typically involves a cost/benefits analysis.

CHAPTER-5

5. SOFTWARE REQURIMENT SPECIFICATION

5.1 Software requirements

Programming Language	python
Operating system	Windows 7, Windows 8 and higher versions, Linux, MacOs
Interpreter	Web application

Table:1.1 Software Requirements

5.2 Hardware requirements

Laptop / PC with 4 GB RAM
Processor – i3 or higher version, AMD 3 or higher version
Storage – 5 GB max
Internet Connection
1 GHz speed

Table:1.2 Hardware requirements

5.3 Hardware interfaces

The solution involves extensive use of several hardware devices. These devices include;

1. Internet modem
2. LAN
3. Windows/Linux/MacOs

5.4 Software interfaces

We are using python programming language for backend programming. Python uses NLTK library for natural language processing to perform the text summarization.

CHAPTER-6

6. SYSTEM DESIGN

6.1 UML Concept

As UML describes the real-time systems, it is very important to make a conceptual model and then proceed gradually. The conceptual model of UML can be mastered by learning the following three major elements −

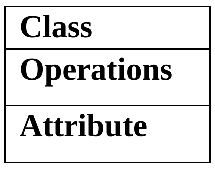
- UML building blocks
- Rules to connect the building blocks
- Common mechanisms of UML

6.2 Building blocks of UML

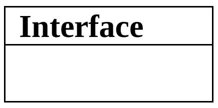
6.2.1 Things in the UML

Structural Things - Structural things define the static part of the model. They represent the physical and conceptual elements. Following are the brief descriptions of the structural things.

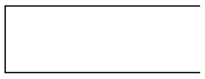
Class − Class represents a set of objects having similar responsibilities.



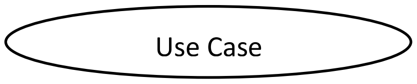
Interface − Interface defines a set of operations, which specify the responsibility of a class.



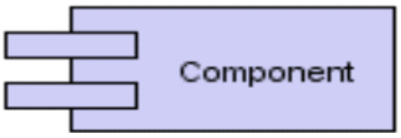
Collaboration −Collaboration defines an interaction between elements.



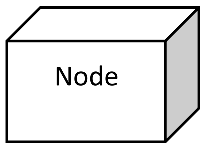
Use case −Use case represents a set of actions performed by a system for a specific goal.



Component –Component describes the physical part of a system.



Node – A node can be defined as a physical element that exists at run time.

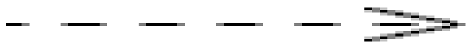


6.2.2 Relationships in UML

Relationship is another most important building block of UML. It shows how the elements are associated with each other and this association describes the functionality of an application. There are four kinds of relationships available.

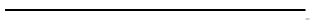
Dependency

Dependency is a relationship between two things in which change in one element also affects the other.



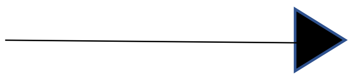
Association

Association is basically a set of links that connects the elements of a UML model. It also describes how many objects are taking part in that relationship.



Generalization

Generalization can be defined as a relationship which connects a specialized element with a generalized element. It basically describes the inheritance relationship in the world of objects.



Realization

Realization can be defined as a relationship in which two elements are connected. One element describes some responsibility, which is not implemented and the other one implements them. This relationship exists in case of interfaces.

------------------------▶

6.2.3 UML diagrams

UML diagram is a diagram that is designed based on Unified Modeling language with the aim to visually represent the system with roles, actors, anchors etc to understand and maintain the system easily. By using this we can better understand flaws or errors in the system so that we can maintain or alter the system properly. Different types of UML diagrams include

1.1 Data Flow diagram

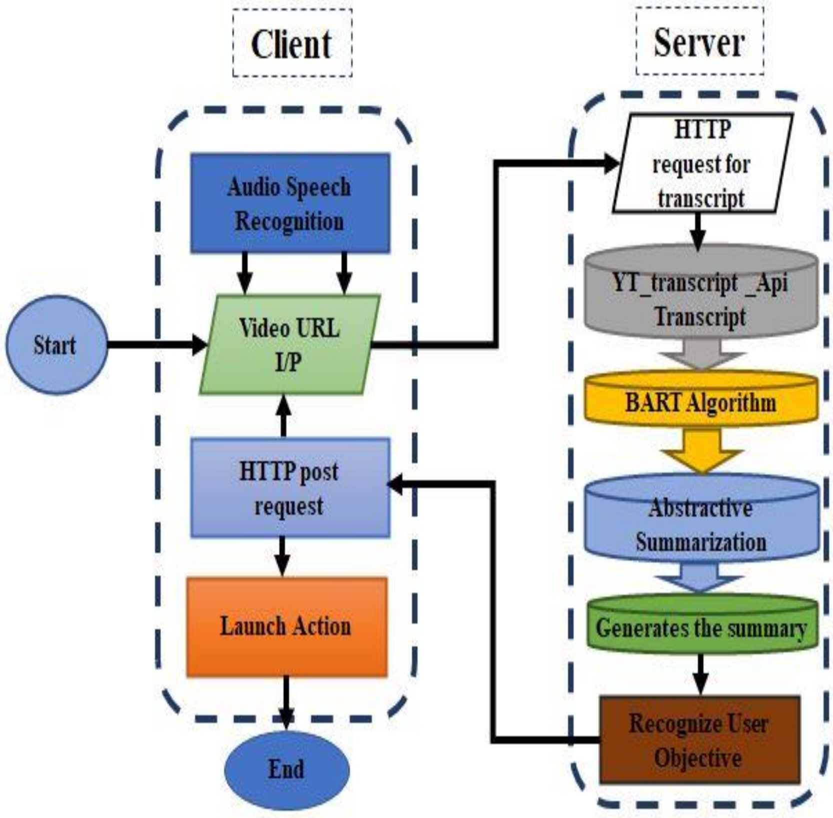


Fig:1.1 Data Flow diagram

1.2 Use case diagram

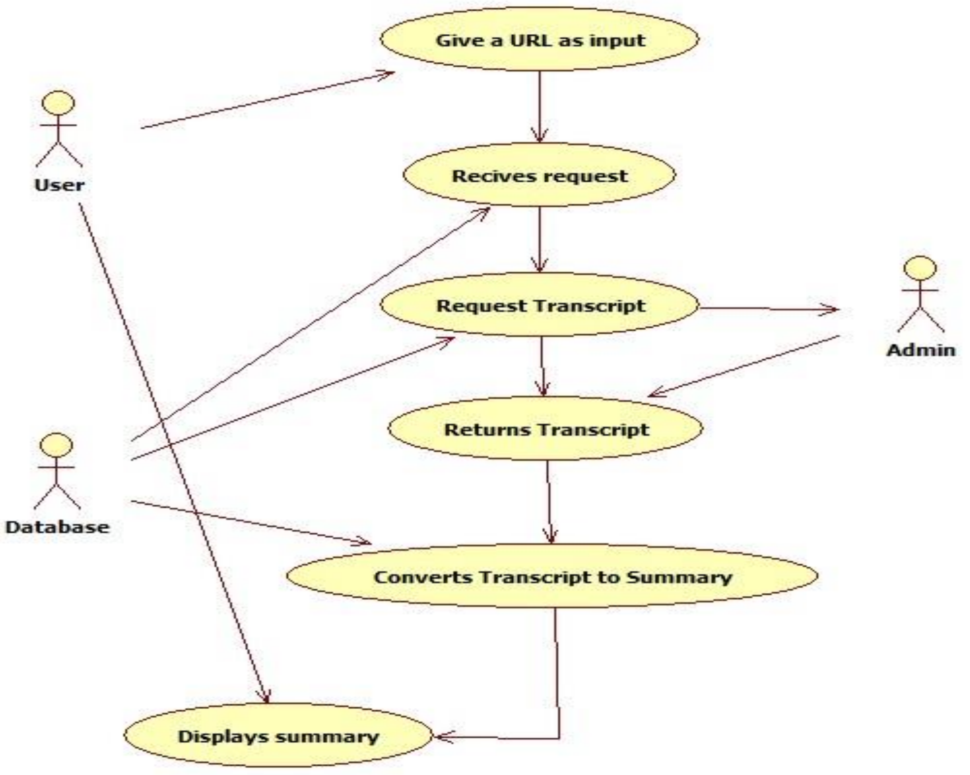


Fig:1.1. Use Case Diagram.

1.3 Class diagram

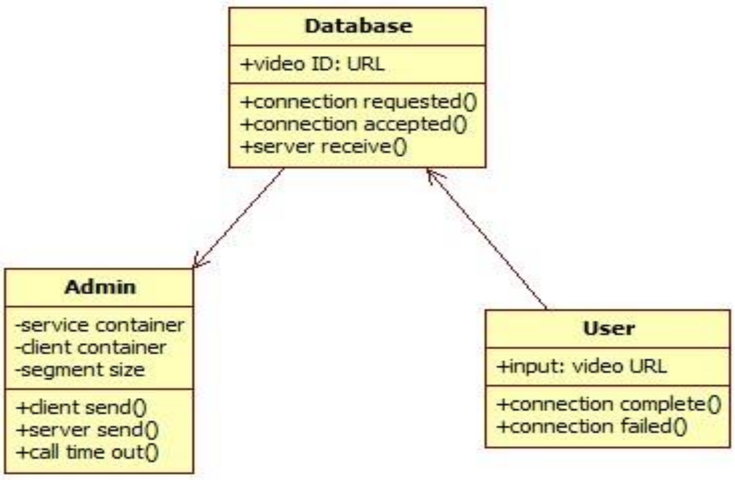


Fig:1.2. Class Diagram.

1.4 Sequence diagram

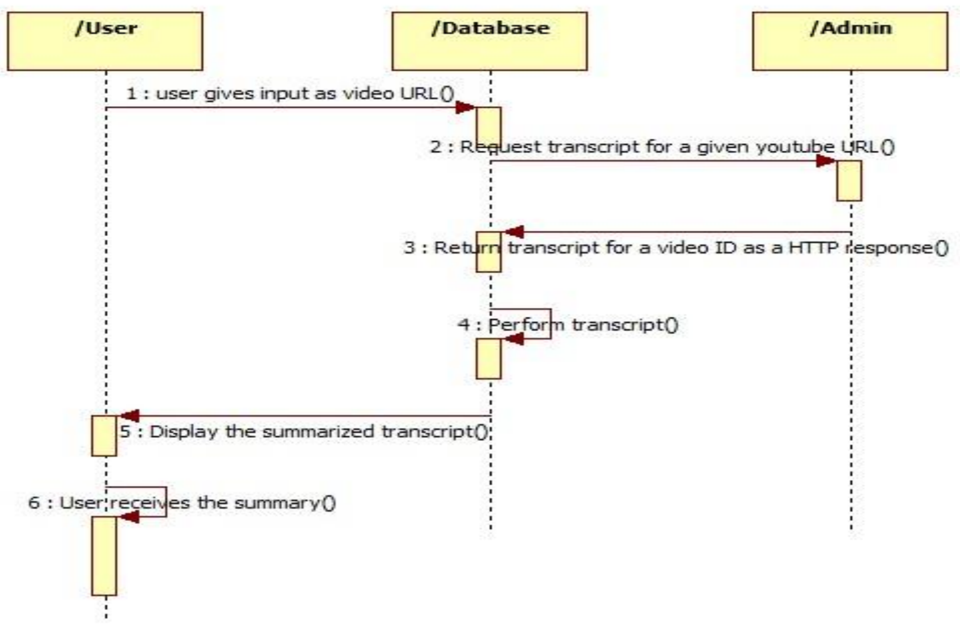

/User
/Database
/Admin
1 : user gives input as video URL()
2 : Request transcript for a given youtube URL()
3 : Return transcript for a video ID as a HTTP response()
4 : Perform transcript()
5 : Display the summarized transcript()
6 : User receives the summary()

Fig:1.3. Sequence Diagram.

1.5 Collaboration diagram

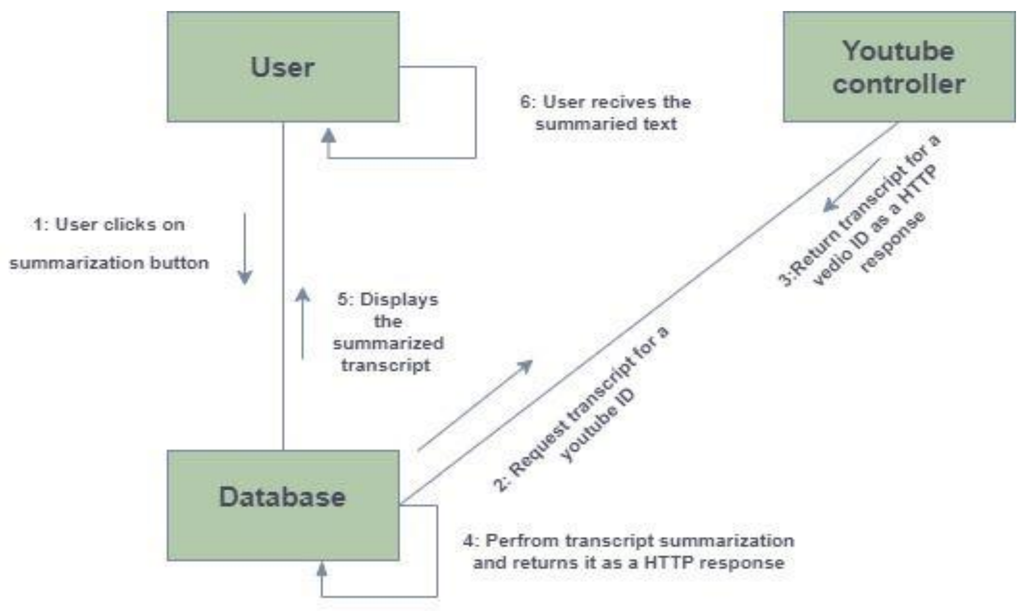

User
Youtube controller
Database
6: User recives the summaried text
1: User clicks on summarization button
5: Displays the summarized transcript
3:Return transcript for a vedio ID as a HTTP response
2: Request transcript for a youtube ID
4: Perfrom transcript summarization and returns it as a HTTP response

Fig:1.4. Collaboration Diagram.

1.6 Activity diagram

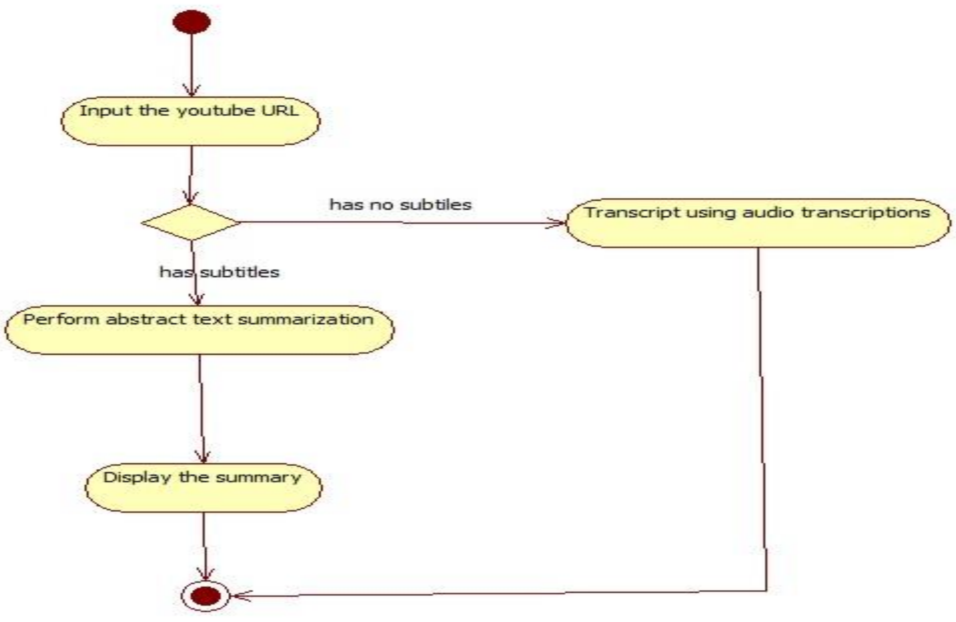

Input the youtube URL
has no subtiles
Transcript using audio transcriptions
has subtitles
Perform abstract text summarization
Display the summary

Fig:1.5. Activity Diagram.

1.7 State Chart diagram

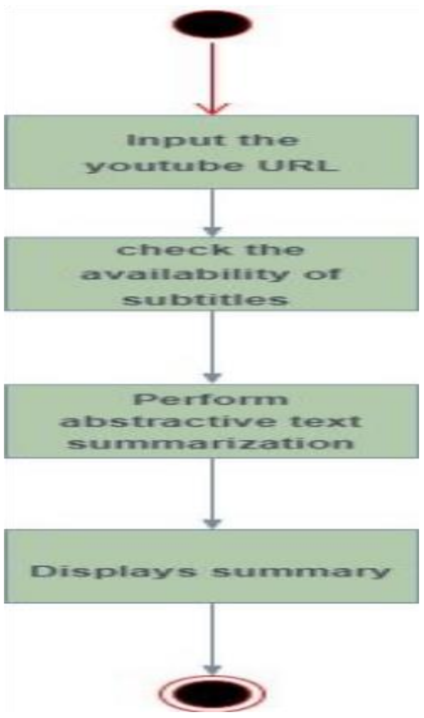

Input the youtube URL
check the availability of subtitles
Perform abstractive text summarization
Displays summary

Fig:1.6. State Chart Diagram.

1.8 Component diagram

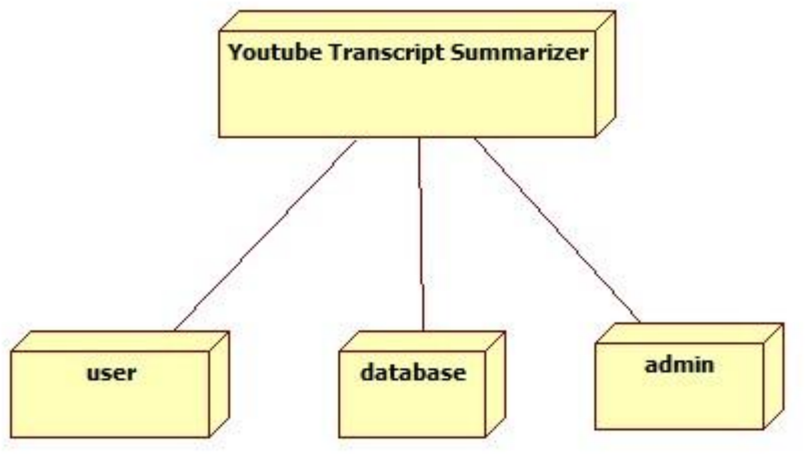


Fig:1.7. Component Diagram.

1.9 Deployment diagram

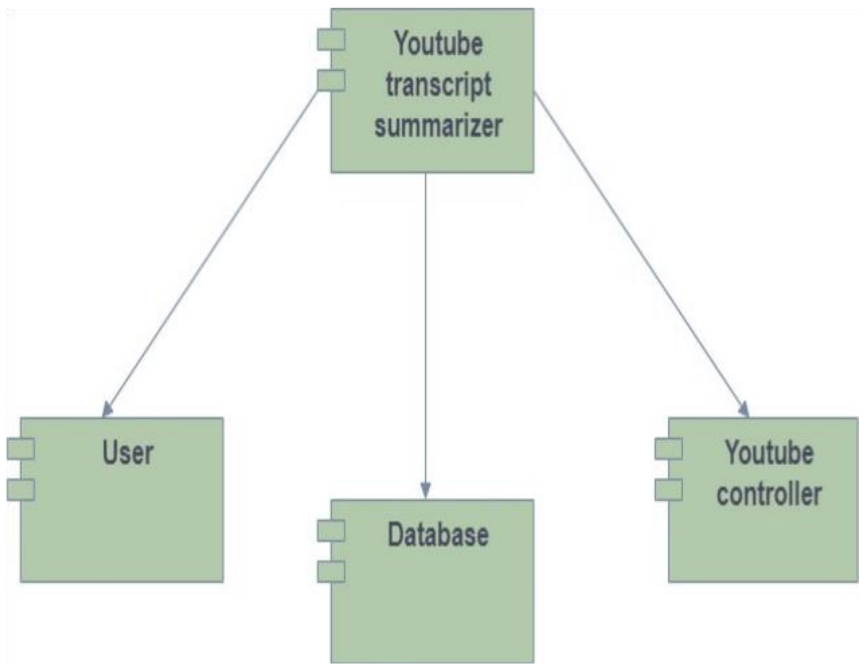


Fig:1.7. Component Diagram.

CHAPTER-7

7. SYSTEM IMPLEMENTATION

7.1 System Architecture

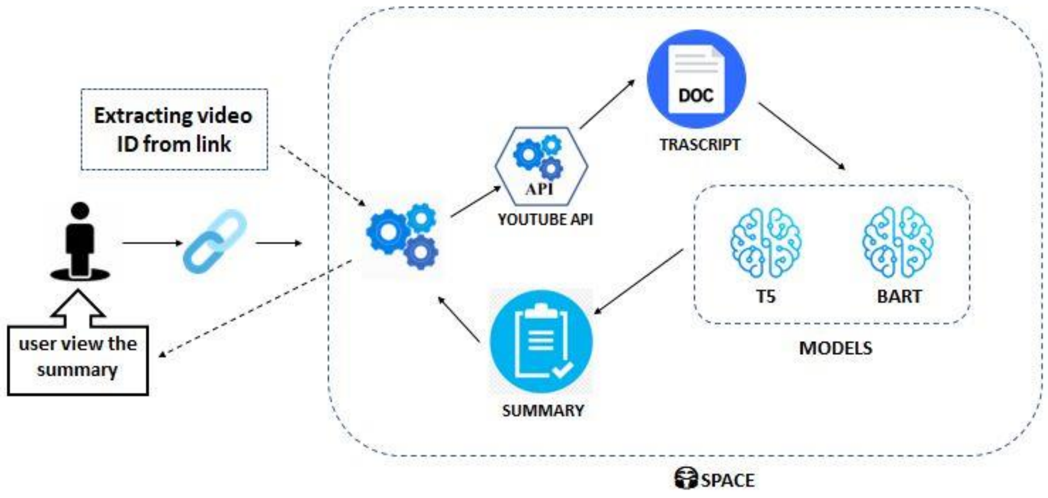


Fig:1.10 Transcript Summarizer Architecture

7.2 Algorithm

7.2.1 BART Model

HuggingFace Transformer models provide an easy-to-use implementation of some of the best performing models in natural language processing. Transformer models are the current state-of-the-art (SOTA) in several NLP tasks such as text classification, text generation, text summarization, and question answering.

The original Transformer is based on an encoder-decoder architecture and is a classic sequence-to-sequence model. The model’s input and output are in the form of a sequence (text), and the encoder learns a high-dimensional representation of the input,which is then mapped to the output by the decoder. This architecture introduced a new form of learning for language-related tasks and, thus, the models spawned from it achieve outstanding results overtaking the existing deep neural network-based methods.

Since the inception of the vanilla Transformer, several recent models inspired by the Transformer used the architecture to improve the benchmark of NLP tasks. Transformer models are first pre-trained on a large text corpus (such as BookCorpus or Wikipedia).

This pretraining makes sure that the model “understands language” and has a decent starting point to learn how to perform further tasks. Hence, after this step, we only have a language model. The ability of the model to understand language is highly significant since it will determine how well you can further train the model for something like text classification or text summarization.

BART is one such Transformer model that takes components from other Transformer models and improves the pretraining learning. BART or Bidirectional and Auto-Regressive

Transformers was proposed in the BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension paper. The BART HuggingFace model allows the pre-trained weights and weights fine-tuned on question-answering, text summarization, conditional text generation, mask filling, and sequence classification.

What is BART Model used for?

As mentioned in the original paper, 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.

Since the unsupervised pretraining of BART results in a language model, 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.

What Data was BART Model Trained on?

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. Just how in computer vision, 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.

So, what exactly counts as noise for text data? The authors of 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.

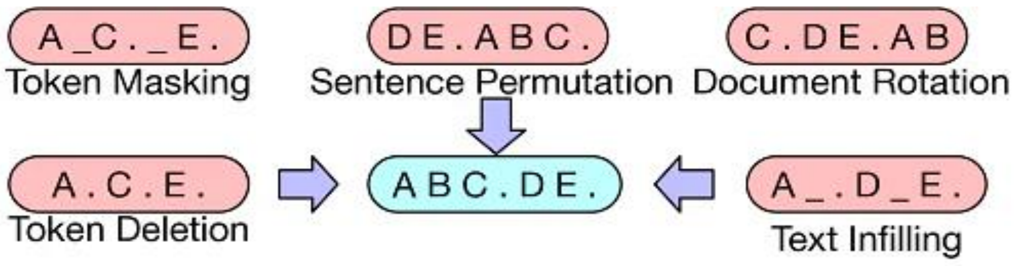


Fig:1.11 BART Model

However, not all transformations are employed in training the final BART model. Based

on a comparative study of pre-training objectives, the authors use only text infilling and sentence permutation transformations, with about 30% of tokens being masked and all sentences permuted.
These transformations are applied to 160GB of text from the English Wikipedia and Book Corpus dataset. With this dataset, the vocabulary size is around 29000, and the maximum length of the sequences is 512 characters in the clean data

7.2.2 BART Algorithm for Summarization.

As more and more long-form content burgeons on the internet, it is becoming increasingly time-consuming for someone like a researcher or journalist to filter out the content they want. Summaries or paraphrase synopsis help readers quickly go through the highlights of a huge amount of textual content and save enough time to study the relevant documents.
Transformer models can automate this NLP task of text summarization. There are two approaches to achieve this: extractive and abstractive. Extractive summarization identifies and extracts the most significant statements from a given document as they are found in the text. This can be considered more of an information retrieval task. Abstractive summarization is more challenging as it aims to understand the entire document and generate paraphrased text to summarize the main points. Transformer models, including BART , perform the latter kind of summarization.

Extractive Summarization

In this type of text summarization, the output is only the important phrases and sentences that the model identifies from the original text.

Abstractive Summarization

For transcript summary extraction you use Abstractive summarization. In this, the output is a different text which is shorter than the original, is generated as a new sentence in a new form by the model. It is just like some human generating summary. In this step, we have used the Encoder-Decoder Seq2Seq model with an attention mechanism to perform abstractive summarization of the text in a transcript after the cleaning process.

Sequence-to-Sequence Model

A Sequence-to-Sequence model takes some sequence of information as an input and generates a sequence of information as an output like mapping a fixed-length input with a fixed-length output where the length of input and output may differ.

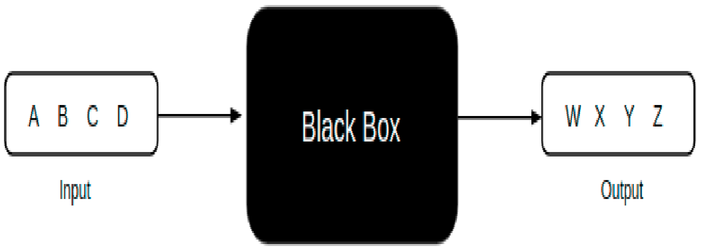


Fig:1.12 Seq2Seq Model

Before giving a sequence of input(words) to the seq2seq model we convert the words into it's vectorized form for that we use some word embedding techniques like Word2Vec, Glove, etc., in our project we have created an embedding layer that converts words into some vector representation so that our model can able to generalize the words to do any kind of prediction or summary generation.

The model composed of an encoder and decoder:

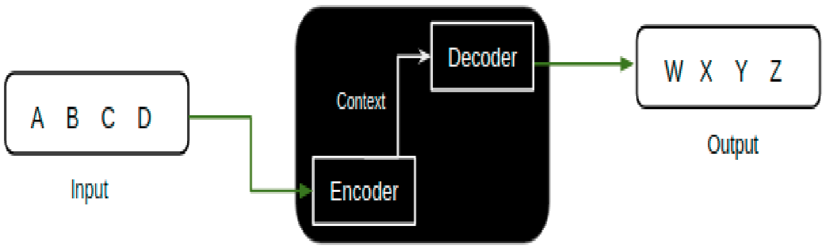


Fig:1.13 Seq2Seq Model

1. Encoder

- ✓ It consists of several recurrent units such as LSTM, which captures the context of the input sequence in the form of a hidden state vector and generates a final embedding at the end of the sequence which is known as context vector, this is then forward to the decoder.
- ✓ In our project, we have used three encoders consisting of recurrent unit LSTM.
- ✓ The hidden state h(i) are computed using the formula:

$$h_t = f(W^{(hh)} h_{t-1} + W^{(hx)} x_t)$$

- ✓ The formula represents the result of LSTM. we tend to simply apply the acceptable weights to the previous hidden state *h(t-1)* and the input vector *x(t).*

Context Vector: Its aim is to encapsulate the data for all input elements so as to assist the decoder make correct predictions.

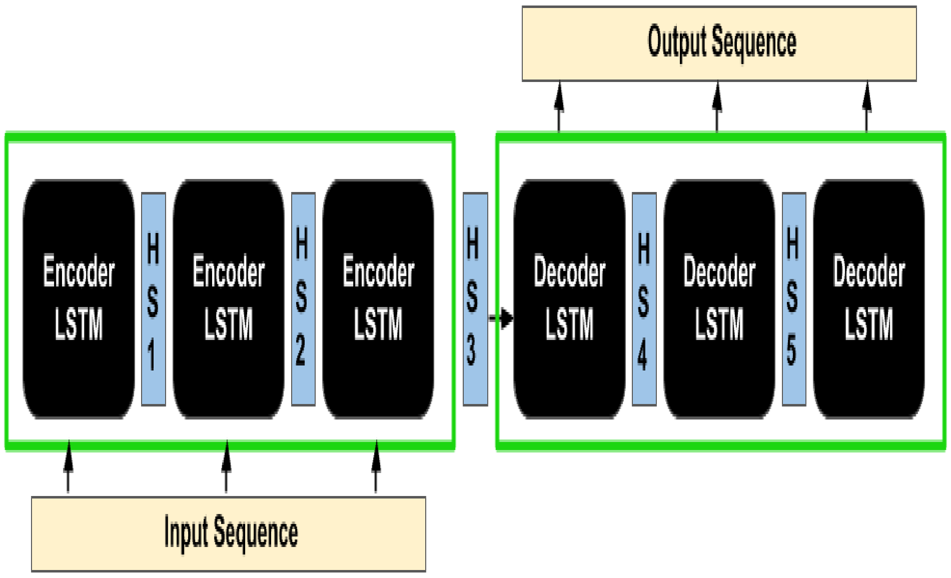


Fig:1.14 Encoder-Decoder Model for Seq2Seq Modelling

2. Decoder

- ✓ A context vector is then sent to the decoder which also consists of recurrent unit LSTM, which then uses it to predict the output sequence y(t) at a time step t, and after each successive prediction, it uses the previous hidden state to predict the subsequent sequence.
- ✓ The hidden sate h(i) is computed using the formula:

$$h_t = f(W^{(hh)}h_{t-1})$$

- ✓ We are using the previous hidden state to work out the subsequent one.
- ✓ The output y(t) at the time step t is calculated using the formula:

$$y_t = softmax(W^S h_t)$$

- ✓ We are calculating the outputs using the hidden state at the present time step alongside the individual weight W(S). Softmax is used to create a probability vector which can facilitate us to determine which word is to incorporate within the summary.

Attention Mechanism

Problem with encoder-decoder is that its output heavily depends on the context defined by the hidden state in the final output of the encoder, since only final layer of encoder generates output which is a context vector, so it makes more challenging for the model to deal with the longer sentences. If any long sequences came in the input, there is a high probability that the initial context has been lost by the end of the sequence. So the answer is to use a method known as "Attention" that permits our model to target completely different components of the input sequence at each stage of output sequence i.e. considering outputs of each encoder layer allowing the context to remain intact from beginning to end. Since a single hidden state vector at the end of the encoder wasn't enough, we sent as many as hidden state vectors. Now the decoder will get the hidden states of all the instances of the input during the phase of decoding.

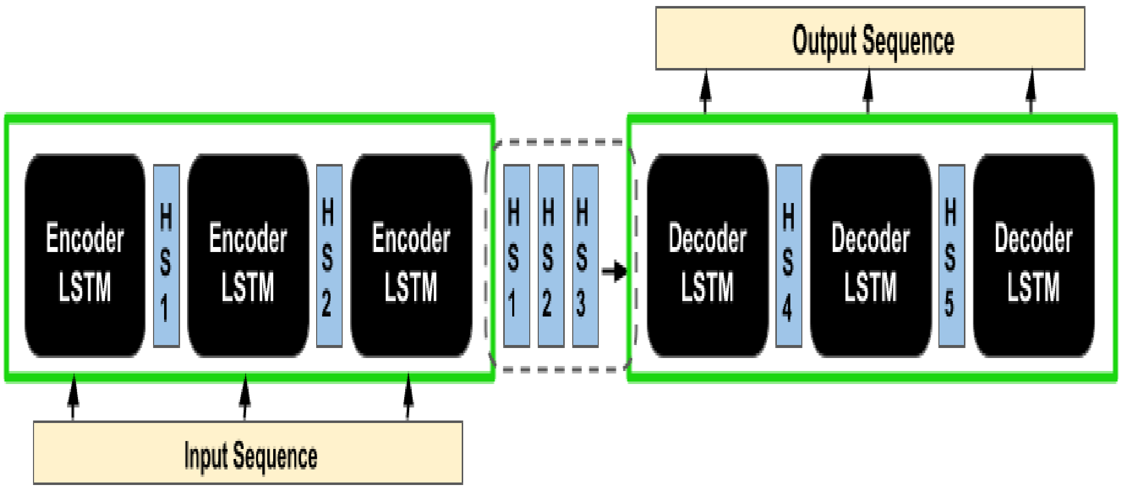
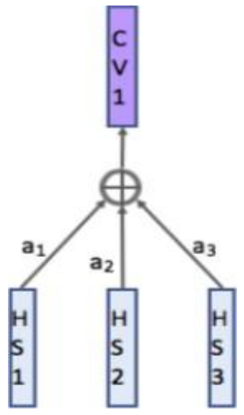


Fig:1.15 Seq2Seq with Attention - Incomplete

Context vector is generated for each time instance within the output sequences. At each step, the context vector is a weighted add of the input hidden states.



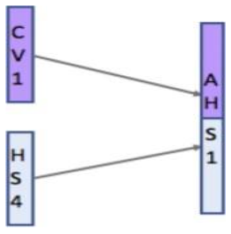


Fig:1.16 Context Vector **Fig:1.17**Attention Hidden State

But how is that the context vector utilized in the prediction? And how a1, a2, a3 weights are decided? The generated context vector is combined with the hidden state vector by concatenation and this new attention hidden vector is employed for predicting the output at that point instance. Note that this attention vector is generated for each time instance within the output sequence and currently replaces the hidden state vector.

Attention scores are the output of another neural network model, the alignment model, that is trained collectively with the seq2seq model at first. The alignment model scores how well an input matches with the previous output and will match this matching for each input with the previous output. Then a softmax is taken of these scores and the resulting number is the attention score for every input.

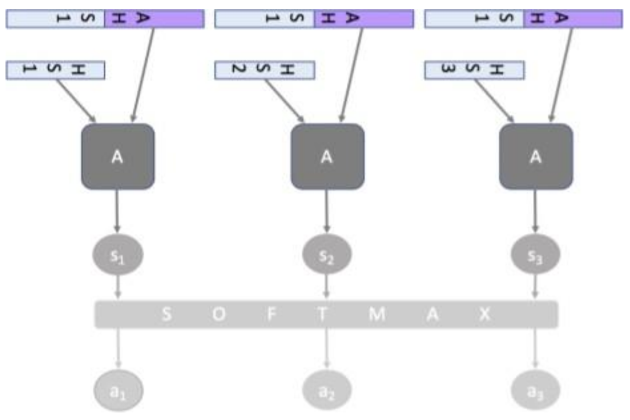


Fig:1.18 Attention Scoring

Hence, we have a tendency to currently apprehend that a part of the input is most vital for the prediction of each of the instances within the output sequence. Within the training section, the model has learned a way to align numerous instances from the output sequence to the input sequence. Below is an illustrated example of a computational linguistics model, shown in an exceedingly matrix kind. Note that each of the entries within the matrix is the attention score related to the input and also the output sequence.

So currently we've got the ultimate and complete model:

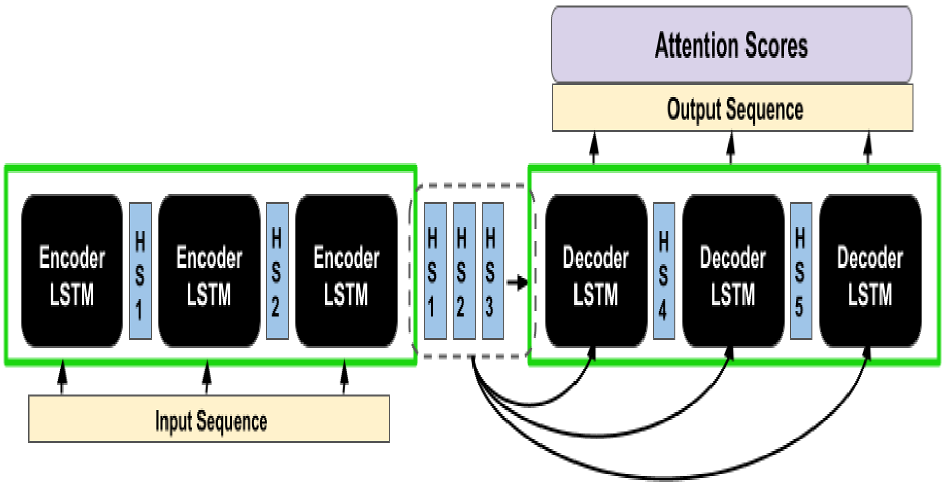


Fig:1.19 Seq2Seq Attention Based Model

Now we are going to train the created model and reserve it for the convenience of the long run use, and using this we will generate our video transcripts summary.

7.3 Source Code

```
!pip install -q transformers
!pip install -q youtube_transcript_api

from transformers import pipeline
from youtube_transcript_api import YouTubeTranscriptApi

youtube_video = https://www.youtube.com/watch?v=1wf8cxiVYW8

video_id = youtube_video.split("=")[1]

video_id

from IPython.display import YouTubeVideo

YouTubeVideo(video_id)

YouTubeTranscriptApi.get_transcript(video_id)
transcript = YouTubeTranscriptApi.get_transcript(video_id)

transcript[0:5]

result = ""
for i in transcript:
    result += ' ' + i['text']
#print(result)
print(len(result))

summarizer = pipeline('summarization')

num_iters = int(len(result)/1000)
```

```
summarized_text = []
for i in range(0, num_iters + 1):
  start = 0
  start = i * 1000
  end = (i + 1) * 1000
  print("input text \n" + result[start:end])
  out = summarizer(result[start:end])
  out = out[0]
  out = out['summary_text']
  print("Summarized text\n"+out)
  summarized_text.append(out)

#print(summarized_text)

len(str(summarized_text))

str(summarized_text)
```

CHAPTER-8

8. SYSTEM TESTING

The main use of testing is to find out errors. Testing is the way toward attempting to find each possible flaw or shortcoming in a work item. It gives a way to deal with check the helpfulness of parts, sub-assemblies, social occasions just as a finished thing It is the path toward working on programming with the point of ensuring that the Software system satisfies its necessities and customer wants and does not bomb in an unacceptable manner. There are various sorts of test. Each test type keeps an eye on a specific testing need. Testing permits to expel the mistakes and improve the framework execution. There are numerous kinds of tests which enables us to improve our venture execution and to make it mistake free. What's more we likewise have tests which encourage us to check singular modules autonomously and furthermore to check complete framework together according to our convenience.

Testing is aimed at ensuring that the system was accurately an efficiently before live operation commands. Testing is best performed when user development is asked to assist in identifying all errors and bugs. The sample data are used for testing. It is not quantity but quality of the data used the matters of testing.

8.1 TYPES OF TESTING

8.1 Unit Testing

Unit testing incorporates the arrangement of analysis that within program basis is working properly, and that program information sources produce significant yields. It checks whether little segments are working appropriately or not. Every single decision branch and inside code stream should be endorsed. It is the attempting of individual programming units of the application . It is done after the completion of an unit before fuse. This is an auxiliary attempting, that relies upon learning of its improvement. Unit tests perform fundamental tests at section level and test a specific business system, application, or possibly structure plan. Unit tests ensure that all of a thoughtful method for a business technique performs unequivocally to the recorded points of interest and contains obviously portrayed data sources and foreseen results. A unit test encourages you to discover which part is broken in your application and fixes it quicker.

8.2 Integration Testing

Integration tests are expected to test joined programming modules to choose whether they everything considered continue running as one program. Testing is an event driven and is dynamically stressed over the crucial after effect of screens or fields. Combination tests show that in spite of the way that the sections were autonomously satisfied, as showed up by successfully unit testing, the gathering of portions are correct and unsurprising. Combination testing is expressly away for revealing the issues that rise up out of the gathering of these portions. Integration testing permits to discover blunders because of unexpected communication between the framework and the sub-framework segments. We test the product in order to test and to identify all the potential mistakes in our undertaking once we complete the source code and before conveying it to the clients. The techniques for performing tests. These techniques provide guidance for testing:

- To test the internal logic of the software components.
- To test the input and output domains of a programs and to uncover the errors in program function, behavior and performance.

We can test the software by using two methods:

8.2.1 Black-Box Testing

Black box testing is a software testing techniques in which functionality of the software under test (SUT) is tested without looking at the internal code structure, implementation details and knowledge of internal paths of the software.

- This type of testing is based entirely on the software requirements and specifications.
- In Black Box Testing we just focus on inputs and output of the software system.

Types of Black Box Testing

There are many types of Black Box Testing but following are the prominent ones -

- Functional testing – This black box testing type is related to functional requirements of a system; it is done by software testers.

- Non-functional testing – This type of black box testing is not related to testing of a specific functionality, but non-functional requirements such as performance, scalability, usability.
- Regression testing – Regression testing is done after code fixes , upgrades or any other system maintenance to check the new code has not affected the existing code.

8.2.2 White-Box Testing

This techniques analyze the internal structures the used data structures, internal design, code structure, and the working of the software rather than just the functionality as in black box testing. It is also called glass box testing or clear box testing or structural testing. White Box Testing is also known as transparent testing, open box testing.

Working process of white box testing

-Input: Requirements, Functional specifications, design documents, source code.
-Processing: Performing risk analysis for guiding through the entire process.
-Proper test planning: Designing test cases so as to cover the entire code. Execute rinse-repeat until error-free software is reached. Also, the results are communicated.
-Output: Preparing final report of the entire testing process.

8.3 System Testing

System testing, also referred to as system-level tests or system-integration testing, is the process in which a quality assurance (QA) team evaluates how the various components of an application interact together in the full, integrated system or application.
System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black-box testing, and as such, should require no knowledge of the inner design of the code or logic. System testing is important because of the following reasons:
System testing is the first step in the Software Development Life Cycle, where the application is tested as a whole.The application is tested in an environment that is very close to the production environment where the application will be deployed. System

testing enables us to test, verify, and validate both the business requirements as well as the application architecture.

SI # Test Case:	STC-1
Name of Test:	System testing in various versions of OS
Item being tested:	OS compatibility.
Sample Input:	Execute the program in windows XP/ Windows-7/8
Expected output:	Performance is better in windows-7
Actual output:	Performance is better in every windows.
Remarks:	Pass

Table:1.3 Test case of STC

8.4 Test cases and Results

SI# Test Case: ¬	UTC-1
Name of Test: ¬	Admin accepts the link.
Items being tested: ¬	Link updated to the Database.
Sample Input: ¬	Video ID, Link.
Expected Output: ¬	Details stored in Database.
Actual Output: ¬	Stored and viewed by Admin.
Remarks: ¬	Pass

Table:1.4 Test case of UTC-1

SI# Test Case: ¬	UTC-2
Name of Test: ¬	User adds video link.
Items being tested: ¬	Verifies the link.
Sample Input: ¬	Video ID, Link.
Expected Output: ¬	Precis text stored in Database.
Actual Output: ¬	Summary.
Remarks: ¬	Pass

Table:1.5 Test case of UTC-2

SI# Test Case: ¬	ITC-1
Name of Test: ¬	Receives URL.
Items being tested: ¬	URL of the video.
Sample Input: ¬	Video URL.
Actual Output: ¬	Summary stored in Database.
Remarks: ¬	Pass

Table:1.6 Test case of ITC-1

SI# Test Case: ¬	ITC-2
Name of Test: ¬	User URL.
Items being tested: ¬	Verifies the transcript from the URL.
Sample Input: ¬	Video ID & URL.
Actual Output: ¬	Summary extracted of the URL.
Remarks: ¬	Pass

Table:1.7 Test case of ITC-2

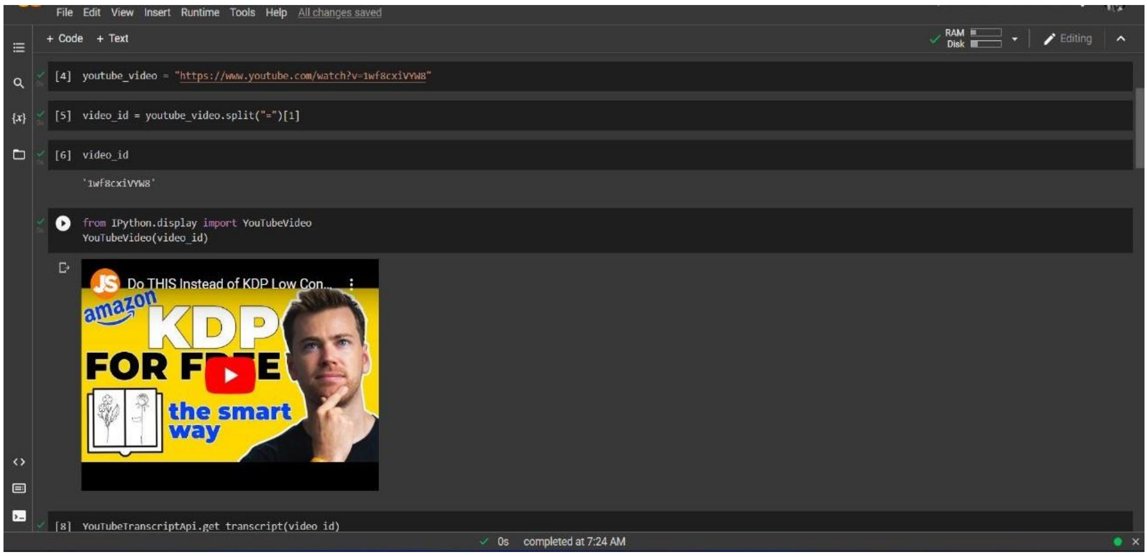
8.5 Verification and Validation

Testing procedure is a piece of subject alluding to checking and approval of our task. We have to find the framework determinations and we should attempt to meet the details of the client and to fulfil the client, for this reason, we need to check and approve the item and we have to ensure that everything is working appropriately. Check and approval are the two unique things. One is performed to guarantee that the product is working accurately and to implement a particular usefulness and the other is done to guarantee if the client prerequisites are appropriately met or not by the finished result. Check is progressively similar to 'would we say we are building the item right?' and approval is increasingly similar to 'would we say we are building the correct item?'

CHAPTER-9

9. OUTPUT SCREENSHOTS AND RESULT ANALYSIS

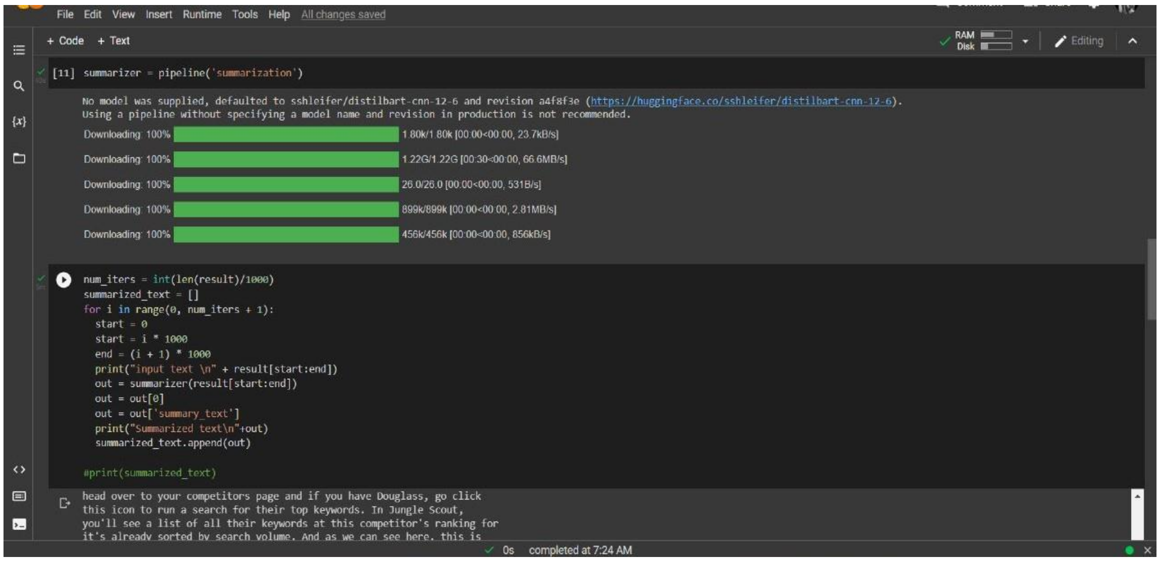
9.1 Input URL


File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
RAM Disk
Editing
Do THIS Instead of KDP Low Con...
amazon KDP FOR FREE the smart way
0s completed at 7:24 AM

```
[4] youtube_video = "https://www.youtube.com/watch?v=1wf8cxiVYW8"

[5] video_id = youtube_video.split("=")[1]

[6] video_id

    '1wf8cxiVYW8'

from IPython.display import YouTubeVideo
YouTubeVideo(video_id)

[8] YouTubeTranscriptApi.get_transcript(video_id)
```

Fig:1.21 User gives the video URL

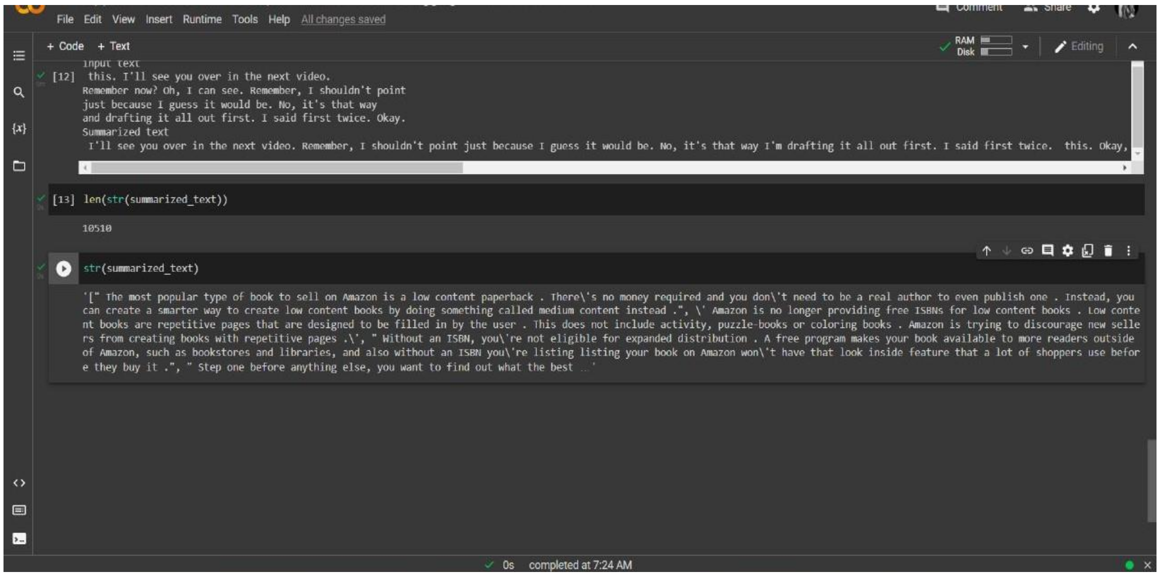
9.2 Transcript Extraction



```
[11] summarizer = pipeline('summarization')

     No model was supplied, defaulted to sshleifer/distilbart-cnn-12-6 and revision a4f8f3e (https://huggingface.co/sshleifer/distilbart-cnn-12-6).
     Using a pipeline without specifying a model name and revision in production is not recommended.
     Downloading: 100%  1.80k/1.80k [00:00<00:00, 23.7kB/s]
     Downloading: 100%  1.22G/1.22G [00:30<00:00, 66.6MB/s]
     Downloading: 100%  26.0/26.0 [00:00<00:00, 531B/s]
     Downloading: 100%  899k/899k [00:00<00:00, 2.81MB/s]
     Downloading: 100%  456k/456k [00:00<00:00, 856kB/s]

num_iters = int(len(result)/1000)
summarized_text = []
for i in range(0, num_iters + 1):
  start = 0
  start = i * 1000
  end = (i + 1) * 1000
  print("input text \n" + result[start:end])
  out = summarizer(result[start:end])
  out = out[0]
  out = out['summary_text']
  print("Summarized text\n"+out)
  summarized_text.append(out)

#print(summarized_text)

head over to your competitors page and if you have Douglass, go click
this icon to run a search for their top keywords. In Jungle Scout,
you'll see a list of all their keywords at this competitor's ranking for
it's already sorted by search volume. And as we can see here, this is
```

Fig:1.22 Generates the transcript of Video URL

9.3 Displays Summary



```
input text
[12]  this. I'll see you over in the next video.
     Remember now? Oh, I can see. Remember, I shouldn't point
     just because I guess it would be. No, it's that way
     and drafting it all out first. I said first twice. Okay.
     Summarized text
      I'll see you over in the next video. Remember, I shouldn't point just because I guess it would be. No, it's that way I'm drafting it all out first. I said first twice.  this. Okay,
```

```
[13] len(str(summarized_text))

10510
```

```
str(summarized_text)

'[" The most popular type of book to sell on Amazon is a low content paperback . There\'s no money required and you don\'t need to be a real author to even publish one . Instead, you
can create a smarter way to create low content books by doing something called medium content instead .", \' Amazon is no longer providing free ISBNs for low content books . Low conte
nt books are repetitive pages that are designed to be filled in by the user . This does not include activity, puzzle-books or coloring books . Amazon is trying to discourage new selle
rs from creating books with repetitive pages .\', " Without an ISBN, you\'re not eligible for expanded distribution . A free program makes your book available to more readers outside
of Amazon, such as bookstores and libraries, and also without an ISBN you\'re listing listing your book on Amazon won\'t have that look inside feature that a lot of shoppers use befor
e they buy it .", " Step one before anything else, you want to find out what the best ...'
```

Fig:1.23 User views the summarized text

CHAPTER-10

10. CONCLUSION

The increase in popularity of video content on the internet requires an efficient way of representing or managing the video. This can be done by representing the videos on the basis of their summary. Learning how to set up web services using API, create Google Chrome extensions and implement the Cloud Computing. In addition, we used HTML and CSS to develop Web Apps and write software packages in python. We need to follow time management and fully grasp the difficulties which may occur and when to change the course of the project based to use our time more efficiently. It is important to understand connections between different technology and to account for possible bugs when incorporating different software packages into the corpus of a final software product.

In this paper, we propose a summarizer and important keywords from the given YouTube video - abstractive.

We have made a simple user interface through which users can easily get their summaries through these methods, and surely find it easy to interact with our user interface and get what they want.

We are sure that our project will surely satisfy the users and solve all the problems that it’s supposed to tackle which is saving time and efforts, by providing only the useful information about the topic which interests them so that they don't have to watch those long videos and the time that saved can be used in gaining more knowledge.

REFERENCE

HuggingFace Transformer:
https://huggingface.co/docs/transformers/installation

XMLHTTP Request:
https://developer.mozilla.org/en
-US/docs/Web/API/XMLHttpRequest/Using_XMLHttpRequest

https://www.crio.do/projects/python-youtube-transcript

Performing Text Summarization:
https://www.thepythoncode.com/article/text-summarization-using
huggingface-transformers-python