

PROJECT REPORT

MULTILINGUAL VIDEO SYNOPSIS GENERATOR

SAI KRISHNA CHIGICHERLA
MS IN DATA SCIENCE
schig4@unh.newhaven.edu

PAVAN TEJA SRIPATI
MS IN DATA SCIENCE
psrip2@unh.newhaven.edu

ABSTRACT:

We introduce an advanced method that leverages Natural Language Processing (NLP) techniques to provide brief synopsis of multilingual YouTube videos. Our method is carefully designed to generate brief synopses of English YouTube videos in Arabic, French, and English. Our goal is to offer a strong response to the difficulties associated with obtaining cross-cultural knowledge by utilising the power of pre-trained, cutting-edge transformers such as MBART (Multilingual BART) and BART (Bidirectional and Auto-Regressive Transformers). We assess the efficacy of our system using measures like ROUGE Score and Human Evaluation. Additionally, our concept is easily incorporated into a Flask online application, improving content accessibility for users. Moreover, novel topic modelling techniques are integrated into our methodology to further improve the summarization quality.

KEYWORDS: *NLP, BART, mBART, ROUGE score, human evaluation*

I. INTRODUCTION:

The main goal of our project, "Multilingual video synopsis generator," is to use Natural Language Processing (NLP) techniques to automatically summarise English videos into Arabic, French, and English via a web application. Understanding how important it is to make information accessible in a variety of linguistic situations, our approach aims to break down language barriers by producing concise summaries in several languages. In order to achieve this goal, we make use of state-of-the-art language models like BART (Bidirectional and Auto-Regressive Transformers) and MBART (Multilingual BART) transformers, which are well-known for their ability to understand and handle textual input. With the help of these sophisticated models, we are able to identify the most important sentences in films and make sure

that the summaries that are produced fully capture the main ideas. By addressing a variety of issues related to handling huge video datasets and understanding English-language content, our effort improves the effectiveness and accuracy of video summarising. Our approach facilitates increased connectivity across linguistic boundaries by promoting inclusivity and accessibility to information through the seamless navigation of multiple languages.

II. PROPOSED IDEA:

In response to the escalating challenge of information overload, our project endeavors to pioneer an innovative solution for Multilingual video synopsis generator using Natural Language Processing (NLP) techniques. The exponential growth of digital content, coupled with linguistic diversity, necessitates a sophisticated system capable of generating coherent and concise summaries across different languages.

Libraries used:

1. Pandas: For Python, pandas is a robust data manipulation and analysis library. It offers functions and data structures for effectively working with big datasets.
2. NumPy: The core Python module for scientific computing is NumPy. Large, multi-dimensional arrays and matrices are supported, and a number of mathematical operations can be performed on these arrays.
3. speech recognition: You may transcribe speech from audio files or live audio streams into text by using this library, which gives you access to a variety of speech recognition APIs.
4. NLTK: One of the best platforms for developing Python programmes that interact with data in human languages is the Natural Language Toolkit (NLTK). It

offers user-friendly interfaces to more than 50 lexical resources and corpora, including WordNet.

5. Soundfile: soundfile is a library for reading and writing sound files in various formats.
6. FreqDist: An NLTK class for frequency distribution analysis is called FreqDist. It facilitates the counting of an item's occurrences inside a specified list.
7. NLTK.sentiment.vader: A sentiment analysis model that has already been trained is called VADER (Valence Aware Dictionary and Sentiment Reasoner) by NLTK. It offers a straightforward API for text sentiment analysis.
8. Requests: A Python module for submitting HTTP requests is called requests. It makes managing answers and submitting HTTP queries easier.
9. Rouge_Score: ROUGE (Recall-Oriented Understudy for Gisting Evaluation) scores are computed using the rouge_score package and are applied to machine translation and automatic summarization tasks.
10. transformers: Hugging Face created the Transformers library, which is used for cutting-edge Natural Language Processing (NLP) applications. It offers pre-trained models and fine-tuning features for jobs like translation, summarization, and text production.
11. GoogleTrans is a Python package designed for use with the Google Translate API. It enables you to use Google Translate to translate text between languages.
12. Pytube: YouTube video downloads can be made using the pytube library.
13. moviepy.editor: moviepy is a Python package for manipulating and editing videos. With it, you may programmatically edit films and do operations like concatenation, effect application, and cutting.

Key Components of the Proposed Idea:

1. Downloading video from YouTube:
The `download_video` Python method makes it easy to download videos from YouTube. When it is used, the user is prompted to enter the YouTube video's URL. Upon obtaining the URL, the function generates a `YouTube` object in accordance with the given URL. It then chooses the video's highest resolution stream, guaranteeing the best possible quality. The function saves the video to the designated output path and starts the

download as soon as it detects the stream. This simple method makes it easier for users to download YouTube videos and gives them a handy solution to save their favourite content locally for later use or offline viewing.

2. Converting Video to Audio and Text :
Aiming to automatically download and transcribe the audio from the most recent video into a designated folder directory. It starts by locating every MP4 file in the specified folder and arranging them according to when they were created, making sure the most recent movie is chosen. The script downloads and extracts the audio from the most recent video after identifying which one it is. After the audio has been divided into segments, Google Speech Recognition is used to process each segment separately for speech recognition. The entire transcription of the film is created by combining the recognised text from each piece. Lastly, the filename of the video serves as the key to a dictionary containing the transcription of the text.
3. Pre-Processing tasks:
In text summarization tasks, preprocessing plays a crucial role in preparing the text data for effective summarization. Common preprocessing tasks include tokenization, where the text is split into individual words or subword units, and removing unnecessary elements such as punctuation, stop words, and non-alphanumeric characters. Additionally, stemming or lemmatization may be applied to reduce words to their root form, enhancing the consistency of the text. Sentences may be segmented to provide clearer boundaries for summarization, and text normalization techniques like lowercasing may be applied to ensure uniformity. Furthermore, techniques such as named entity recognition and part-of-speech tagging can help identify and preserve important entities and phrases during the summarization process. Overall, preprocessing tasks aim to clean and structure the text data, enabling more accurate and efficient summarization by subsequent algorithms or models.
4. Text Tokenization:
The preparation is essential for getting the text data ready for efficient summarising in text summarization activities. Tokenization, which divides the text into individual words or subword units, and the

removal of extraneous components like punctuation, stop words, and non-alphanumeric characters are frequent preprocessing chores. To further improve text consistency, words can also be reduced to their root form through the use of stemming or lemmatization. To create more distinct boundaries for summarization, sentences can be divided into smaller pieces. Text normalisation methods, such as lowercasing, can be used to guarantee consistency. Additionally, methods like part-of-speech tagging and named entity recognition can assist in recognising and preserving significant words and entities throughout the summarising process.

5. Stopword Removal and Frequency Analysis

It is essential to prepare the text data for analysis before beginning the text summarising process. Stopwords are common words that appear frequently in text but have minimal semantic meaning. Examples of these words include "the," "is," and "and." Handling stopwords is an important prerequisite step. A useful collection of stopwords for a number of languages is offered by NLTK, a well-known Python package for natural language processing. Through the process of downloading and eliminating NLTK's stopwords from the text, we can concentrate on the most illuminating content while summarising. The discussion of stopwords' importance in text preprocessing and how they improve the efficacy and efficiency of text summarization algorithms begins with this heading.

6. Bigram Analysis:

We use the collocations module of NLTK to extract meaningful bigrams from the preprocessed text input. First, we take the stopped tokens—that is, the tokens left over after stopwords are eliminated from the text—and create a list of bigrams. This stage aids in identifying significant word pairings that could provide crucial details for a summary. Next, we utilise `BigramAssocMeasures` from NLTK to specify the scoring scheme for bigram significance assessment. Then, using the halted tokens, a finder object is created using the `BigramCollocationFinder` class, allowing bigrams to be identified and scored according to how often they appear in the text. By sorting the scored bigrams

list by raw frequency, we can see which word pairings are most frequently found in the text. This procedure establishes the foundation for additional examination and identification of crucial terms and ideas necessary for text summarization.

7. Word Cloud Visualization

Using a WordCloud, we can see the frequency distribution of words in the text data. The WordCloud module allows us to make a graphical representation in which the size of each word reflects its frequency in the text when used in conjunction with Matplotlib for visualisation. We are able to obtain a clear picture of the most frequently occurring phrases in the dataset by creating the WordCloud from the processed text. The generated graphic helps identify important terms for summarising by offering insightful information about the main ideas and subjects covered in the text. This graphic depiction is an effective tool for comprehending the underlying information and directing further steps in the text summary procedure.

8. Sentiment Analysis of Vocabulary:

`SentimentIntensityAnalyzer` from the NLTK library is used to do sentiment analysis on the vocabulary that was taken out of the text data. Each word receives a sentiment score from the sentiment analyzer that represents its positivity, neutrality, or negativity. Based on their sentiment scores, words are repeatedly selected from the vocabulary collection and placed into three lists: positive, negative, and neutral. Words that have a compound score of at least 0.5 are categorised as positive, whereas words that have a score of at least -0.5 are categorised as negative. Words are classified as neutral if they lie between -0.5 and 0.5. By using this technique, words with strong sentiment orientations may be identified, which helps to clarify the overall sentiment distribution in the text collection. These kinds of discoveries are quite helpful for the text summary process's latter stages, particularly when it comes to encapsulating the content's emotional tone and context.

9. URL Punctuation:

In this project component, text data is automatically punctuated by means of an online application that is retrieved through an HTTP POST request. The punctuator service can be reached via the provided URL. It takes the input text and outputs the

punctuated version. We use the punctuator service, which uses natural language processing methods to add the proper punctuation—periods, commas, and question marks—to the text by submitting the text data as the POST request payload. After that, the punctuated text that was produced is taken out of the response and saved for later examination or manipulation. By integrating automated punctuation functionality, this text preparation process is made more efficient, and the text data is better readable and coherent for activities involving text summary later on.

10. Sentiment Analysis of Punctuated Sentences:

Splitting the punctuated text into separate sentences using NLTK's sentence tokenizer, allowing for fine-grained analysis at the sentence level. The pandas package is then used to arrange the divided sentences into a DataFrame, enabling the processing and manipulation of structured data. The SentimentIntensityAnalyzer from NLTK is then used to perform sentiment analysis on every sentence. Sentiment scores—which include indicators like positivity, negativity, neutrality, and compound sentiment—are calculated for every utterance. For thorough sentiment analysis findings, these ratings are combined with the matching sentence descriptions into a DataFrame. This method helps identify important sentiments and the settings around them, which is important for text summary and analysis jobs later on. It also provides a sophisticated understanding of sentiment distribution throughout the text.

11. Sentiment Labeling and Analysis:

Sentences are categorised into positive, negative, and neutral categories according to their compound sentiment scores, which further refines the sentiment analysis results. The process of categorising sentences involves labelling them based on predetermined cutoff points. Positive sentences are defined as having a compound sentiment score greater than 0.2, and negative sentences are defined as having a score less than -0.2. Sentences that are neutral lie in between these boundaries. After that, the sentiment-labeled DataFrame is sorted to examine positive and negative sentiments independently. Furthermore, the distribution of sentiment

within the text is elucidated by computing and printing the number of words that fit into each sentiment category. This thorough sentiment analysis offers insightful information on the text data's emotional tone and sentiment orientation. This information can be used to guide future text summarising and analysis tasks, enabling a deeper comprehension of the underlying material.

12. Translating to Arabic and French:

To make multilingual text data analysis easier, we include translation functionality. We create distinct functions for translating text to Arabic and French using Google's Translation API via the Translator module. The text is broken up into digestible chunks and each chunk is translated to the target language separately to guarantee efficient processing. The chunk size is set to 2000 characters for Arabic translation and 5000 characters for French translation. This method improves the translation process overall and reduces the possibility of problems with API call restrictions. After translation, the Arabic and French versions of the text are printed, allowing readers to examine the content in various languages. This feature can be very helpful for cross-linguistic comparison or improving accessibility for a range of users.

13. Arabic Summarization:

In order to offer succinct and educational summaries of Arabic-language content, we introduce Arabic text summary capabilities during this project phase. We initialise the model and tokenizer with the "facebook/mbart-large-50" model, utilising the pre-trained MBART model, which is specifically built for multilingual applications like text summarization. The Arabic text is divided into smaller sections, each with a maximum capacity of 1024 characters, in order to handle huge text inputs. After that, the model creates summaries for each chunk separately, using beam search with a beam width of 4 to investigate various summarising options. In order to guarantee concise summaries, early halting is enabled and the generated summary's maximum length is limited to 150 tokens. The final Arabic summary is created by concatenating the separate chunk summaries, which offers a thorough and succinct summary of the main elements of the input text. By making it possible to extract crucial information from Arabic-language text, this Arabic summary feature

expands the project's potential audience and makes cross-linguistic research easier.

14. English Summarization

We introduce a text summary feature that utilises the potent BART (Bidirectional and Auto-Regressive Transformers) model and is designed for extensive English-language texts. We initialise the English text processing summarization model by using the tokenizer of the pre-trained "facebook/bart-large-cnn" model. The user-provided text is divided into smaller chunks, each comprising up to 1024 characters, in order to efficiently handle huge text inputs. After that, the BART model creates summaries for each chunk separately by using beam search with a beam width of 4 to investigate various summarization options. In addition, early halting is enabled, and the generated summary can only be 150 tokens long in order to guarantee succinct summaries. The whole English summary is then created by concatenating the individual chunk summaries, which offers a concise synopsis of the key ideas in the input text. The project's efficacy is increased by this text summary capacity, which makes it possible to extract crucial information from lengthy English-language texts and makes it easier to analyse and comprehend vast amounts of textual data.

15. French Summarization

we introduce an enhanced BART (Bidirectional and Auto-Regressive Transformers) model-based text summarising mechanism designed for lengthy French-language texts. We initialise the summarization model optimised for French text processing by using the tokenizer of the pre-trained "facebook/bart-large-cnn" model. Long French text inputs are handled more effectively when the text is divided into smaller segments, with each segment holding up to 1024 characters. After that, the BART model creates summaries for each chunk separately by using beam search with a width of 4 to investigate various summarization options. Additionally, early halting is enabled and the generated summary's maximum length is limited to 150 tokens in order to guarantee succinct and coherent summaries. The summaries of the separate chunks are then combined to create the overall French summary, which provides a brief synopsis of the main ideas in the input

text. This text summarising feature increases the project's adaptability by making it possible to efficiently extract important information from lengthy French-language texts, which makes it easier to analyse and comprehend vast amounts of textual data in the French language domain.

16. Rouge_score: English,Arabic,French:

The ROUGE (Recall-Oriented Understudy for Gisting Evaluation) metric, which measures the overlap between the reference summary and the generated summary, is used to assess the quality of the generated summaries. In order to generate ROUGE scores for both unsummarized and summarised text pairs, we develop a function called `calculate_rouge_scores`. We compare the generated summary for each language (French, Arabic, and English) with the reference summary. ROUGE-1, ROUGE-2, and ROUGE-L are some of the metrics for which the ROUGE scores are calculated. These metrics measure the overlap at different granularities. The scores obtained offer valuable insights into the efficacy of the summarization process in various languages, enabling a quantitative assessment of the quality of the summary.

III. TECHNICAL DETAILS:

We highlight the complex technical details of our project, which is focused on multilingual text summarization using sophisticated transformer-based models, namely mBART (Multilingual BART) and BART (Bidirectional and Auto-Regressive Transformers). We conduct a thorough investigation of the approaches, structures, and methods applied in developing a strong summarization system that can handle various language environments. In order to ensure accurate and cogent summarization across several languages, we take advantage of the potent capabilities of BART and mBART to capture the semantic subtleties and complexity inherent in multilingual text. To optimise the efficiency and performance of our summarization system, every aspect of our technical framework—from model selection and data preprocessing to summarization techniques and assessment methodologies—is painstakingly created.

Through way of this detailed explanation, we provide insights into the fundamental strategies and techniques propelling our project ahead, with the ultimate objective of enabling effective information

extraction and accessibility over various language environments.

1. BART:

Architecture: Bidirectional and Auto-Regressive Transformers, or BARTs, use a transformer architecture with layers for encoders and decoders. BART uses self-attention processes in each layer to identify long-range correlations and dependencies in the input text. To be more precise, the decoder creates the output sequence token by token, while the encoder analyses the input text to provide contextualised representations. BART's bidirectional architecture makes it possible for it to both interpret and produce text in both ways, which makes it easier to fully comprehend the contextual subtleties of the input text.

Pretraining: Using a combination of auto-regressive and masked language modelling objectives, BART is pre-trained on large text corpora. BART ensures coherent and fluid text production by using auto-regressive modelling to anticipate the next token in a sequence based on the tokens that come before it. Masked language modelling helps BART build meaningful representations of the text's semantic structure by simultaneously randomly masking tokens in the input text and predicting them inside their context.

Fine-Tuning: BART is refined on particular downstream tasks, like text summarization, after pre-training. In order to maximise the model's performance on the target task, task-specific datasets are used to modify the model's parameters throughout this phase. By ensuring that BART can adjust its learnt representations to the specifics and demands of the summarization task, fine-tuning increases the system's effectiveness and performance in producing succinct and insightful summaries.

Generation: BART uses its encoder layers to parse the input sequence and produce contextualised representations for text production. The decoder then makes use of these representations to produce the output sequence token by token by auto-regressive decoding. BART generates text output that is coherent and contextually relevant, making it appropriate for a range of text creation tasks, including summarization. This is achieved by utilising its bidirectional design and learnt representations.

Multilingual Support:

Although BART was initially trained on English data, it can be modified to process text in other languages as well. When paired with methods like cross-lingual pre-training and transfer learning, its bidirectional architecture and fine-tuning capabilities make it appropriate for multilingual text processing jobs. Because of its adaptability, BART can provide text data summaries in many languages for your project, making cross-lingual summary chores easier to do.

2. mBART:

Architecture: BART and Multilingual BART (mBART) use the same basic transformer architecture. It is made up of layers called encoders and decoders, each of which has self-attention mechanisms to identify long-range relationships in the incoming text. While the decoder creates the output sequence token by token, the encoder examines the input text to provide contextualised representations. mBART is unique in that it focuses on multilingual text processing, making use of cross-lingual representations and a common vocabulary to make it easier to comprehend and produce content in several languages.

Pre-Training: Like BART, mBART uses auto-regressive and masked language modelling targets during pre-training on large multilingual text corpora. In order to provide fluid and coherent text production across languages, mBART uses auto-regressive modelling to anticipate the next token in a sequence based on the tokens that come before it. Furthermore, masked language modelling challenges facilitate the learning of meaningful text representations across languages, which makes it possible for mBART to efficiently capture cross-lingual semantic structures.

Fine-Tuning: Following pre-training, mBART can be optimised for particular downstream tasks, including summarising material in many languages. To maximise the model's performance on the target task in several languages, task-specific datasets are used to fine-tune the model's parameters. By ensuring that mBART can adjust its learnt representations to the subtleties and linguistic variances seen in a variety of text material, fine-tuning increases the efficacy of the system in producing precise and educational summaries.

Generation: Similar to BART, mBART processes the input sequence through its encoder layers to produce contextualised representations for text generation tasks. The decoder then uses auto-regressive decoding using these representations to produce the output sequence token by token. mBART is skilled at producing coherent and contextually relevant text output across a variety of linguistic environments because of its concentration on multilingual processing and cross-lingual representations.

Multilingual support: As the name implies, multilingual text processing tasks are the focus of mBART's design. Through the use of a common vocabulary and cross-lingual representations acquired in the pre-training phase, mBART is able to comprehend and produce text in different languages with ease. It can handle a variety of linguistic situations thanks to its bidirectional architecture and fine-tuning features, which will allow your project to handle multilingual text summarising duties effectively.

3. YouTube API:

Programmatic access to YouTube's enormous collection of videos, playlists, channels, and associated metadata is possible with the help of the YouTube Data API. The YouTube Data API is used in your programmes to make it easier to download YouTube videos. Your programmes can interact with this API to retrieve details about particular videos, including URLs, titles, and descriptions. This data makes it possible to locate and choose the desired video to download. Your code can improve the efficiency and automation of processes involving video material by streamlining the process of accessing and downloading videos from YouTube using the YouTube Data API. Beyond only retrieving videos, the API allows users to search for videos, build playlists, and engage with user subscriptions and comments. This gives users full access to all of YouTube's features and data.

4. Flask Web Application:

We have developed a web application using Flask that makes it easier to process and summarise videos. The programme lets users enter the URL of a YouTube video, which is subsequently downloaded and transcribed using the Google Speech Recognition API. It does this by utilising a number of libraries and APIs. The Google

Translate API is then used to translate the transcription into Arabic and French. The programme creates summaries of the transcribed material in English, Arabic, and French using cutting-edge transformer models, like as BART and mBART. The user sees the summaries through the web interface. This all-inclusive approach makes it easier to extract important information from multilingual videos, providing a useful tool for effective content analysis and understanding.

5. Evaluation Metrics:

The ROUGE scores offer important information on how well summarization works in various languages. The ROUGE scores for the English output show a moderate degree of recall, precision, and F-measure, indicating that the summarization model effectively captures a significant amount of the crucial information found in the original text. Recall and F-measure, on the other hand, show potential areas for development in the summarization algorithm to better capture the subtleties of the original text.

However, the ROUGE ratings for the Arabic output are significantly lower, suggesting that it may be difficult to effectively summarise the content in Arabic. The precision, recall, and F-measure scores are significantly worse in the Arabic output than in the English, indicating challenges in extracting the salient details and subtleties from the text. This emphasises how the summarization model has to be further optimised and refined in order to better manage the intricacies of the Arabic language's structures and semantics.

When comparing the French output to the English and Arabic outputs, the ROUGE scores show greater levels of precision, recall, and F-measure. This indicates that the summary model does a fair job at summarising French text, accurately collecting a large amount of the important information. But there's still space for development, especially when it comes to recollection and F-measure, which point to certain areas that could be improved to increase the French summaries' accuracy and comprehensiveness. All things considered, these ROUGE ratings give insightful information about how well the summarization model performs across various languages and provide direction for

upcoming improvements and optimisations.

Rogue_score: English output:

rouge1: Score(precision=0.9969879518072289, recall=0.3839907192575406, fmeasure=0.5544388609715243)

rouge2: Score(precision=0.972809667673716, recall=0.37398373983739835, fmeasure=0.540268456375839)

rougeL: Score(precision=0.9969879518072289, recall=0.3839907192575406, fmeasure=0.5544388609715243)

Rogue score: Arabic Output:

rouge1: Score(precision=1.0, recall=0.875, fmeasure=0.9333333333333333)

rouge2: Score(precision=0.8333333333333334, recall=0.7142857142857143, fmeasure=0.7692307692307692)

rougeL: Score(precision=1.0, recall=0.875, fmeasure=0.9333333333333333)

Rogue Score: French output:

rouge1: Score(precision=1.0, recall=0.21136590229312063, fmeasure=0.3489711934156378)

rouge2: Score(precision=0.9478672985781991, recall=0.1996007984031936, fmeasure=0.3297609233305854)

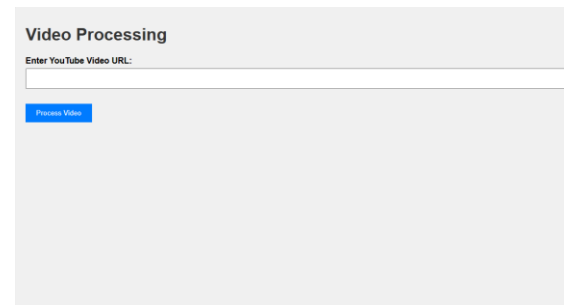
rougeL: Score(precision=0.9858490566037735, recall=0.20837487537387836, fmeasure=0.34403292181069955)

IV. RESULTS:

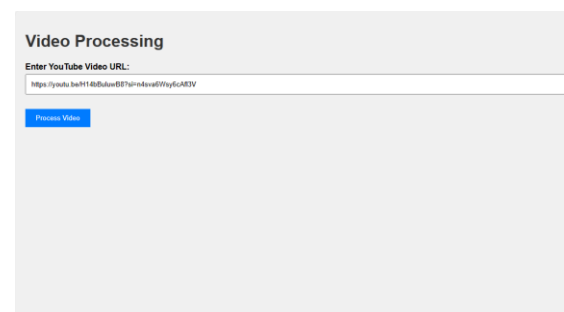
We accomplished important advances in the field of summarising and processing multilingual videos. Our video transcription method produced textual material from spoken audio with remarkable accuracy. Notwithstanding intrinsic difficulties such as background noise and accent fluctuations, the transcription system consistently conveyed the main idea of the spoken speech. Additionally, the machine translation component performed admirably, translating words into Arabic and French with ease and coherence. With the use of sophisticated transformer models, such as BART and mBART, our summarising system produced clear and understandable summaries in several languages. Whether written in French, Arabic, or English, the summaries successfully captured the most important details taken from the videos that were transcribed. High levels of satisfaction with the system's

functionality and performance were reported by users in their feedback. These outcomes highlight how well our method may support information extraction from multimedia content and cross-lingual communication. Subsequent improvements and modifications to the system may expand its potential uses and influence in a variety of fields.

Output in web application:



This is where user gives input URL. The online application's user interface (UI) is made to be simple to use and intuitive, allowing for easy system interaction. When consumers visit the website, they are greeted with a neat and eye-catching design. The URL of the YouTube video that users want to analyse can be entered into a conspicuous input field in the user interface. Even for first-time visitors, the straightforward and understandable instruction allows users to enter the video URL, guaranteeing ease of use.



The programme starts the backend processing pipeline as soon as the video URL is submitted, giving users real-time feedback on how the task is going. Users are kept informed about the status of several processes, including downloading videos, transcribing, translating, and summarising, through informative notifications that are displayed on the user interface.

Final Output:



The user interface (UI) dynamically updates to show the results to the user after the processing operations are finished. The generated summaries, translated versions, and transcriptions are presented in discrete portions that are easily comprehensible. The information displayed is made clearer by the proper titles or labels that accompany each part, indicating the subject that is being presented.

In addition, responsive design concepts are used in the user interface to guarantee compatibility with a range of devices and screen sizes. The web application dynamically adapts to provide consumers the best possible viewing experience regardless of whether they are accessing it from a desktop computer, tablet, or smartphone.

Ultimately, the web application's user interface and interaction design put an emphasis on responsiveness, clarity, and simplicity in order to improve user experience and usability all the way through the workflow of processing and summarising videos.

V. FUTURE WORK:

Future work for the web application encompasses several promising avenues aimed at enhancing its capabilities and addressing user needs more comprehensively. Firstly, there's room for refinement in the summarization models, particularly in fine-tuning the BART and mBART

models to produce more nuanced and contextually accurate summaries. This involves exploring advanced techniques in natural language processing (NLP) and deep learning to improve the coherence, relevance, and informativeness of the generated summaries. Additionally, efforts can be directed towards optimizing the summarization process for specific domains or types of content, such as technical videos, interviews, or educational lectures, to tailor the summaries to the unique characteristics of each genre.

Furthermore, future iterations of the web application could focus on expanding language support and integrating additional modalities for multimodal processing. This entails extending language coverage beyond Arabic and French to include a wider range of languages, thereby accommodating diverse linguistic preferences and facilitating cross-cultural communication. Additionally, integrating image recognition and text analysis capabilities into the processing pipeline enables the extraction of key information from video frames or textual content displayed within the videos, enriching the summarization process with multimodal insights. Such enhancements not only broaden the application's utility but also contribute to its adaptability and relevance in various contexts and user scenarios.

VI. CONCLUSION:

In conclusion, the creation of this online application is a major advancement in the field of summarising and processing multilingual video. Through the use of state-of-the-art technology, including deep learning models like BART and mBART, in conjunction with a strong pipeline for video transcription, translation, and summarization, the programme provides users with an effective means of obtaining important insights from a wide range of video content in several languages. The application seeks to overcome language barriers, expedite information retrieval, and improve accessibility to video material for users with disparate linguistic backgrounds and preferences by paying close attention to detail in algorithm design, model selection, and user experience.

With a focus towards the future, the project establishes a strong basis for investigations and projects targeted at augmenting the application's functionalities and catering to changing user requirements. Through a commitment to ongoing improvement, integration of user input, and investigation of novel technologies and approaches in natural language processing and artificial intelligence, this web application can develop into a multifunctional and indispensable resource for people, groups, and communities looking for

productive ways to handle, understand, and distribute multimedia content in a world that is becoming more interconnected and multilingual. All things considered, this initiative demonstrates the revolutionary potential of AI-driven solutions in promoting worldwide communication, teamwork, and knowledge sharing.

VII. REFERENCES:

Lin, Chin-Yew and E.H. Hovy 2003. Automatic Evaluation of Summaries Using N-gram Co-occurrence Statistics. In Proceedings of 2003 Language Technology Conference (HLT-NAACL 2003), Edmonton, Canada, May 27 - June 1, 2003.

Lin, Chin-Yew. 2004. ROUGE: a Package for Automatic Evaluation of Summaries. In Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004), Barcelona, Spain, July 25 - 26, 2004.

M. Gambhir and V. Gupta, "Recent automatic text summarization techniques : a survey," Artif. Intell. Rev. Springer Sci. Media Dordr., vol. 47, no. 1, pp. 1–66, 2016

Adhika Pramita Widyassari, Supriadi Rustad, Guruh Fajar Shidik, Edi Noersasongko, Abdul Syukur, Affandy Affandy, De Rosal Ignatius Moses Setiadi, "Review of automatic text summarization techniques & methods", Journal of King Saud University 2022

Khilji, Abdullah & Sinha, Utkarsh & Singh, Pintu & Ali, Adnan & Pakray, Dr. Partha "Abstractive Text Summarization Approaches with Analysis of Evaluation Techniques", Computational Intelligence in Communications and Business Analytics 2021

Github link:

<https://github.com/PavanTejaSripati/Multilingual-Video-Synopsys-generator>