PROJECT REPORT

ON

OPTIMIZED VIDEO TEXT ANALYZER

Submitted for fulfillment of award of

Bachelor of Technology

In

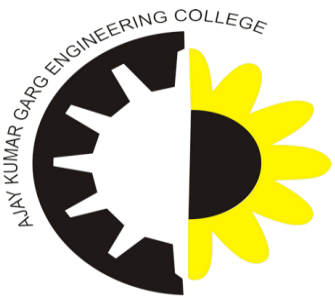
Computer Science & Information Technology

By

UMENDRA MANI DWIVEDI	**1900270110059**
ISHITA VASHISTHA	**2000270119004**
SHIVAM SINGH	**1900270130159**
TUSHAR RAGHAV	**1900270110057**

Under the Guidance of

Dr. Anupama Sharma



Ajay Kumar Garg Engineering College Ghaziabad

(2022-23)

CERTIFICATE

This is to certify that the project report entitled **Optimized Video Text Analyzer** submitted by **Umendra Mani Dwivedi, Ishita Vashistha, Shivam Singh, and Tushar Raghav** to the Ajay Kumar Garg Engineering College Ghaziabad in partial fulfillment for the award of the degree of **B. Tech in (Information Technology/Computer Science & Information Technology)** is a *bona fide* record of project work carried out by him/her under my supervision. The contents of this report, in full or in parts, have not been submitted to any other Institution or University for the award of any degree or diploma.

Dr Anupama Sharma

Supervisor

Department of Information Technology

Counter signature of HOD with seal

DECLARATION

We hereby declare that this project report titled **Optimized Video Text Analyzer** submitted in partial fulfillment of the degree of **B. Tech in (Information Technology/Computer Science & Information Technology)** is a record of original work carried out by me under the supervision of **Dr. Anupama Sharma, Associate Professor Department of Information and Technology**, and has not formed the basis for the award of any other degree or diploma, in this or any other Institution or University. In keeping with the ethical practice of reporting scientific information, due acknowledgments have been made wherever the findings of others have been cited.

Umendra Mani Dwivedi
1900270110059

Ishita Vashistha
2000270119004

Shivam Singh
1900270130159

Tushar Raghav
1900270110057

ACKNOWLEDGMENTS

We take this opportunity to thank **Dr Anupama Sharma mam and MS Mrignainy mam and Mr. Pancham sir who helped in preparing the report.**

We extend my sincere thanks to one and all of the AKGEC family for the completion of this document on the project report format guidelines.

Umendra Mani Dwivedi
Ishita Vashistha
Shivam Singh
Tushar Raghav

ABSTRACT

Optimized Video Text Analyzer is a smart tool that is used to conduct in-depth analysis on the user-uploaded video. This article is to represent the working and the result analysis of proposed and implemented Optimized Video Text Analyzer. The main objective of the Optimized Video Text Analyzer is to extract metadata from the required video, which will be used to get the precise, optimized and summarized information from the video related to search, event-detection, or classification. It is to identify structure and extract features from the raw video to get semantically meaningful representations and summarized information from the video such that time and cost of the user can be reduced. It should be possible to automatically extract structural and semantic content from video domains without any involvement of humans. Algorithms for text analysis can be used to summarize the entire video or convert a text summary and also that summary into audio format, as well as to identify the key points covered in the video and contents that contain solutions to a specific problem.

Optimized video text analyzer uses the combined technique of Deep Learning and Natural Language Processing in which the concept of Convolutional Neural Network is used for Image processing and Recurrent Neural Network for the summarization of extracted text from the video respectively. After extraction of text data, cleaning and preprocessing of data is performed so that the model could be easily trained over that data and the result must be produced with greater accuracy and efficiency. Then creation of the model will be performed with the help of CNN and RNN. The optimized video text analyzer works more effectively and efficiently than a standard video text analyzer because it offers more helpful features and functionalities like highlighting the key points.

TABLE OF CONTENTS

DESCRIPTION **PAGE NUMBER**

LIST OF FIGURES

ABBREVIATIONS/ NOTATIONS/ NOMENCLATURE

CNN	Convolutional Neural Network
LSTM	Long Short Term Memory
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
OCR	Optical Character Recognition
RNN	Recurrent Neural Network
SRS	Software Requirement Specification

`

CHAPTER 1
INTRODUCTION

1.1 PREFACE

In the current digital era, technological development is advancing quickly, which is generating a huge volume of video data every day. Nowadays, social media is one of the most used applications on a large scale in our daily lives. One of the media types that is used most frequently on social media is video.

Video data contains beneficial textual information such as scene text and caption text. The different types of videos like movies, news videos, and TV programs video etc. are created by various video frames based on its purpose. The majority of videos are distributed via social media, and this one can be repeated or lack vital information. However, the consumer is simply left with the choice of downloading the movie or watching the entire thing. This leads to add more and more of the costs to users because the video requires a large bandwidth to download video or view it and a large space to store it and most important it is very time consuming for the user because they to watch the full video for getting only some sort of required information. Thus, it is important to be a way to enable users to watch videos in a way that helps to reduce time and costs.

Optimized video text analyzer technique is a proposed solution to resolve different points such as power consumption and additional cost. The processing of such huge chunky videos requires high storage, high computational processing power, and consumes a lot of time. Extraction of features from the video is a time-consuming task because the user has to watch the entire video. A large number of editing tools exist that require expertise that is highly expensive. The optimized video text analyzer is used to overcome these issues as it produce summaries by analyzing the underlying content of a source video stream, condensing this content into abbreviated descriptive forms that represent surrogates of the original content embedded within the video. The multimodal nature of video, which conveys a wide range of semantics in multiple modes, such as sound, music, still images, moving image, and text, makes this task much more complex than analyzing text documents.

`

1.2 MOTIVATION OF THE STUDY

The field of video analysis and video to text summarization is an area of active research in both the information retrieval and natural language processing communities. Video content is becoming increasingly popular on various online platforms, and extracting valuable information from it has become crucial for various industries. In order to achieve this goal, an optimized video text analyzer is required, which can extract textual information from videos and then convert it into audio to produce a summary in audio format. This will help users to easily comprehend the content of the video without having to watch the entire video.

The primary aim of an optimized video text analyzer is to obtain metadata from the uploaded video, which will be useful in the creation of a summary of the whole video. This summary can then be used to get important information and required features from the videos that can be helpful for the user. For example, in the education industry, a video text analyzer can be used to summarize the content of educational videos for students who may have trouble understanding the content. In the entertainment industry, it can be used to analyze the content of various videos and help businesses decide which videos to promote.

The process of summarizing video content involves several steps, including extracting textual information from the video, cleaning and preprocessing the data, and then using machine learning algorithms to create a summary of the content. An optimized video text analyzer can save time, effort, and resources, while also improving the accessibility and comprehensibility of the video content.

1.3 PURPOSE

The purpose of an optimized video text analyzer project is to develop a tool that can automatically analyze the text within videos and provide useful insights to users. This project aims to improve the accuracy and efficiency of various industries, such as video editing, media monitoring, education, marketing, and law enforcement.

`

The primary purpose of this project is to develop an optimized video text analyzer that can accurately identify and analyze the text within videos. This tool should be able to identify key words and phrases, analyze the tone of the content, and generate useful insights for users.

By developing an optimized video text analyzer, we can help to streamline various processes across different industries. For example, in video editing, this tool can help editors to quickly identify key points and create highlights of those points in the final video. In media monitoring, this tool can help to track the usage of specific keywords or phrases in videos, which can help to identify potential trends or issues. In education, this tool can help educators to create more engaging and interactive video content. In marketing, it can help to optimize video content for search engines and improve the overall performance of video marketing campaigns. In law enforcement, this tool can help to identify suspects, track criminal activity, and provide evidence in court.

The purpose of this project is to create a valuable tool that can be used across multiple industries to improve efficiency and accuracy. By developing an optimized video text analyzer, we can help to revolutionize the way that we analyze and interpret video content, and provide valuable insights to users.

1.4 OBJECTIVE AND SCOPE

The objective of optimized video text analyzer is to extract text from video frames with high accuracy and efficiency. This technology has a wide range of applications in fields such as security, healthcare, and education, among others. The scope of optimized video text analyzer includes developing algorithms, models, and techniques that can accurately and efficiently extract text from video frames.

The primary objective of optimized video text analyzer is to enable automated text extraction from video frames. This is achieved through the use of deep learning algorithms, specifically convolutional neural networks (CNNs), which are designed to analyze and process image data. By using CNNs, optimized video text analyzer algorithms are able to detect and extract text from video frames with high accuracy.

`

The scope of optimized video text analyzer extends beyond just text extraction. It also includes the development of technologies that can preprocess video data, such as image enhancement and segmentation, to improve the quality of the input data. Additionally, optimized video text analyzer algorithms can be combined with natural language processing (NLP) techniques to enable more advanced applications, such as automatic translation of video text into different languages.

In terms of applications, optimized video text analyzer has a wide range of potential use cases. In security, it can be used to automatically analyze video feeds from surveillance cameras and extract relevant text, such as license plate numbers or facial recognition information. In healthcare, optimized video text analyzer can be used to automatically extract text from medical images and improve diagnosis accuracy. In education, it can be used to automatically transcribe lectures and presentations, making them more accessible to students with disabilities.

In summary, the objective of optimized video text analyzer is to enable automated text extraction from video frames, while the scope includes the development of algorithms, models, and techniques to achieve this objective. The technology has a wide range of applications in various fields, and as it continues to evolve, it is likely to become an even more valuable tool for automated video analysis and processing.

1.5 BACKGROUND

The paragraph discusses the technologies and libraries used in a particular project. The first part of the paragraph, i.e., section 1.5.1 to 1.5.3, talks about the technologies utilized in the project. It is important to mention the technologies used in a project as it gives an insight into the technical knowledge and expertise of the project team. The second part of the paragraph, i.e., section 1.6, talks about the libraries used in the project. A library is a collection of pre-written code that can be reused for specific functionalities, thus reducing the development time and effort required. Libraries can also improve the efficiency and quality of a project.

`

Overall, mentioning the technologies and libraries used in a project can help others understand the approach and tools used in the project and can also be useful for future references or improvements.

1.5.1 Natural Language Processing

Natural Language Processing (NLP) is a subfield of Computer Science that deals with the interactions between computers and natural language. It aims to develop algorithms and computational models that can help computers understand, interpret, and generate human language. One of the important tasks in NLP is text summarization, which involves producing a shortened version of a given text while retaining its most important information. Since computers are not capable of understanding human language in the same way as humans do, they need to be trained on large amounts of data and use various NLP techniques to identify important information in the text and produce a summary that is readable and understandable to humans.

NLP is closely related to the field of Artificial Intelligence (AI) because it involves developing intelligent systems that can understand and generate human language. Many existing AI algorithms and techniques, including neural networks, are used in NLP to develop models that can learn from data and perform tasks such as sentiment analysis, machine translation, and text summarization. Researchers generally rely on two types of approaches for text summarization: extractive and abstractive summarization. Extractive summarization involves selecting the most important sentences or phrases from the original text and combining them to form a summary. Abstractive summarization, on the other hand, involves generating new sentences that capture the essence of the original text, which can be more challenging but also more effective in producing summaries that are concise and informative.

1.5.2 RNN and LSTM

Recurrent Neural Networks (RNNs) were developed to address this challenge of sequence learning. RNNs work by processing each input element sequentially, maintaining a hidden state that carries information about the preceding elements in the sequence. The output from the current element is then dependent on the current input element and the

`

hidden state from the previous element. This hidden state serves as the memory of the network and allows it to model sequences of arbitrary length.

However, standard RNNs have a limitation called the vanishing gradient problem. This problem arises when gradients in the backpropagation algorithm become very small, making it difficult to learn long-term dependencies. Long Short-Term Memory (LSTM) networks were introduced to address this issue.

LSTMs are a type of RNN that uses memory cells to store information for an extended period. These memory cells can add or delete information from the cell state using gates that control the flow of information. The three gates used in LSTMs are the input gate, forget gate, and output gate. The input gate decides which values to update in the cell state, the forget gate decides which values to remove from the cell state, and the output gate decides which values to output based on the current cell state.

LSTMs have been shown to be effective in tasks that require modeling long-term dependencies, such as speech recognition, language translation, and image captioning. They are particularly useful in natural language processing tasks such as sentiment analysis, where the context of a sentence is critical for determining its meaning. The use of LSTMs allows the model to capture the context and use it to make accurate predictions.

Overall, RNNs and LSTMs are powerful tools for modeling sequences and have found widespread use in natural language processing, speech recognition, and other applications that require modeling temporal data.

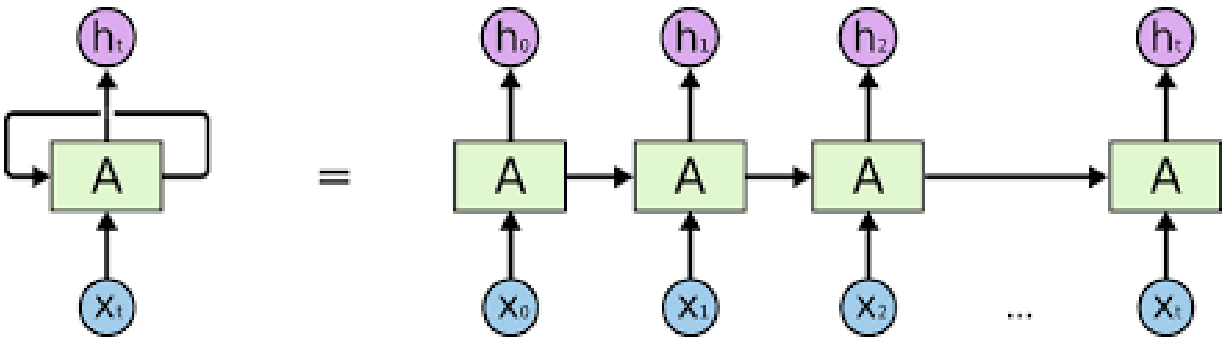


Figure 1: An unrolled recurrent neural network

`

Figure 1 depicts the structure of a recurrent neural network (RNN) when it is unrolled. The diagram shows the sequence of input data being fed into the network at different time steps, with each time step represented by a column in the diagram. In this diagram, "ht" represents the output units value after each timestamp, "x" represents the input units, and "A" represents a chunk of the neural network. The figure shows that the result from the previous timestamp is passed to the next step for part of the calculation that occurs in a chunk of the neural network. In this way, the network can capture information from the previous timestamp, which is critical for tasks like text summarization, where the sequence of words in input documents is important.

However, traditional RNNs have limitations in effectively memorizing information when the distance between connected information increases. This is because each activation functions is nonlinear, making it challenging to trace back to hundreds or thousands of operations to retrieve information. For instance, a sentence that is too long for the network to memorize may lead to errors in the output of the model. In this scenario, Long Short-Term Memory (LSTM) networks can be used to address the limitations of traditional RNNs. LSTM networks have an additional memory cell that can maintain information over a longer period. The cell's state can be written, read, and erased using special structures called gates. This mechanism allows the LSTM network to convey information in the long term while avoiding the vanishing gradient problem that occurs in traditional RNNs.

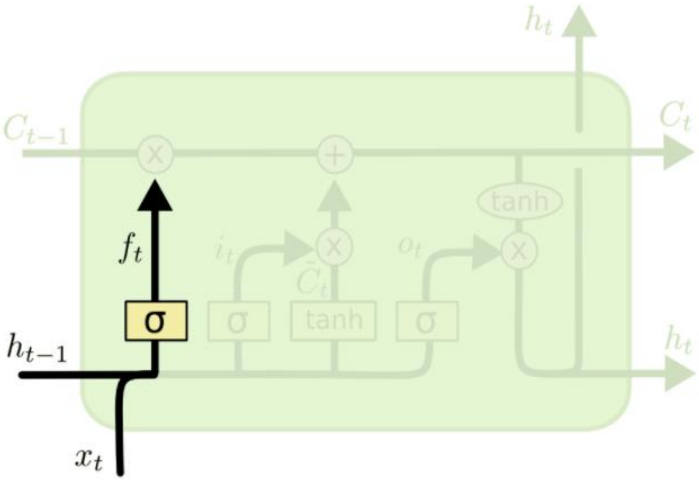


Figure 2: The linear operation in LSTM cell

`

Different from the traditional RNN, inside each LSTM cell, there are several simple linear operations which allow data to be conveyed without doing the complex computation. As shown in Figure 3, the previous cell state containing all the information so far smoothly goes through an LSTM cell by doing some linear operations. Inside, each LTSM cell makes decisions about what information to keep, and when to allow reads, writes and erasures of information via three gates that open and close.

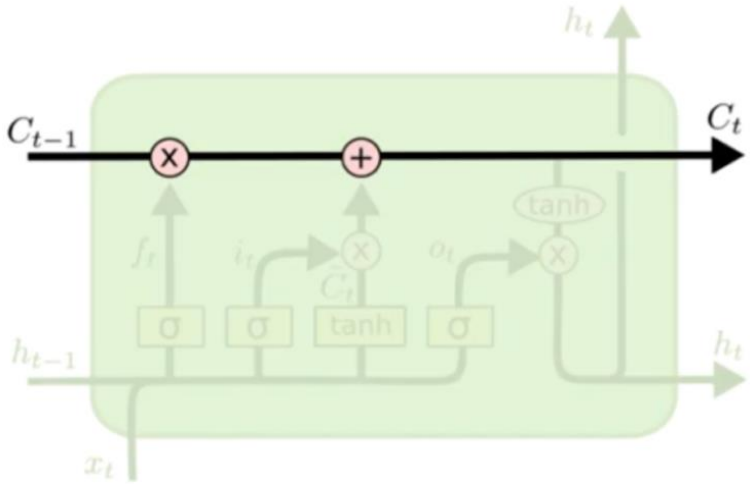


Figure 3: The forget gate layer

As shown in Figure 3, the first gate is called the “forget gate layer”, which takes the previous output units value ht-1 and the current input xt, and outputs a number between 0 and 1 to indicate the ratio of passing information. 0 means do not let any information pass, while 1 means let all information pass. To decide what information needs to be updated, LSTM contains the “input gate layer” as shown in Figure 4. It also takes in the previous output units value ht-1 and the current input xt and outputs a number to indicate inside which cells the information should be updated. Then, the previous cell state Ct-1 is updated to the new state Ct. The last gate is “output gate layer”, which decides what the output should be.

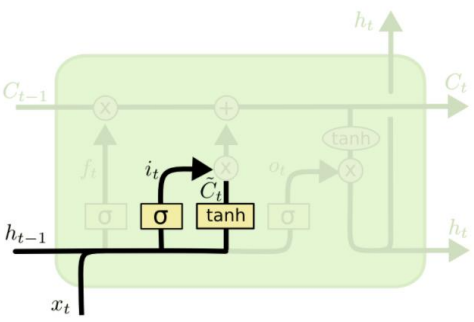


Figure 4: The input gate layer

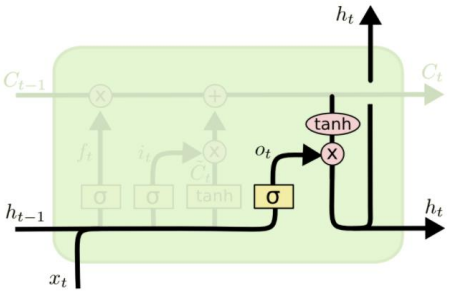


Figure 5: The output gate layer

`

Figure 5 shows that in the output layer, the cell state is going through a tanh function, and then it is multiplied by the weighted output of the sigmoid function. So, the output units value ht is passed to the next LSTM cell (Christopher, 2015). Simple linear operators connect the three gate layers. The vast LSTM neural network consists of many LSTM cells, and all information is passed through all the cells while the critical information is kept to the end, no matter how many cells the network has.

1.6 LIBRARIES

1.6.1 Keras

The Keras library is a Python library that provides a high-level API for building and training deep learning models. In an optimized Video Text Analyzer, the Keras library can be used in the following ways:

- Text Classification: The Keras library can be used to build deep learning models for text classification tasks such as sentiment analysis, topic modeling, and named entity recognition. These models can be trained on the preprocessed text data to classify it into different categories or labels.

- Video Captioning: The Keras library can be used to build deep learning models for video captioning tasks. These models can be trained on the preprocessed video frames and the extracted text to generate captions for the video.

- Text Generation: The Keras library can be used to build deep learning models for text generation tasks such as chatbots and language translation. These models can be trained on large text datasets to generate text based on the input.

- Transfer Learning: The Keras library provides several pre-trained models for text and image processing tasks. These models can be fine-tuned on the preprocessed text and video data to improve the accuracy of the analyzer.

Overall, the Keras library can be a valuable tool in an optimized Video Text Analyzer,

`

allowing for the creation of deep learning models for various text and video processing tasks.

It can also help to improve the accuracy and efficiency of the analyzer by leveraging pre-trained models and transfer learning.

1.6.2 Natural Language Toolkit

NLTK (Natural Language Toolkit) is a popular Python library used for natural language processing tasks. It provides a set of tools and algorithms to work with human language data, such as tokenization, stemming, and part-of-speech tagging.
These features can be utilized in an optimized video text analyzer in the following ways:

- Text Preprocessing: Before analyzing the text from the video, it is important to preprocess the text to remove unwanted characters and normalize the text. NLTK provides various tools for tokenization, sentence segmentation, and stemming that can be used to preprocess the text.

- Named Entity Recognition: NLTK provides algorithms for identifying named entities in text, such as people, organizations, and locations. By using this feature, the video text analyzer can identify important entities mentioned in the video and perform sentiment analysis on them.

- Part-of-Speech Tagging: NLTK provides tools for identifying the part of speech of each word in a text. This feature can be used to identify the subject, verb, and object of the sentence, which can be useful for analyzing the context of the text.

- Sentiment Analysis: NLTK provides tools for performing sentiment analysis on text. This feature can be used to analyze the sentiment of the text in the video, which can be useful for understanding the overall tone of the video.

In an optimized video text analyzer, NLTK library can be used to perform several language processing tasks on the text extracted from a video. This can enable the analyzer to analyze the context and sentiment of the video, which can be useful for a variety of applications, such as content moderation, video recommendation, and customer feedback analysis.

`

For instance, the video text analyzer can use NLTK to identify and classify the language used in the video as positive, negative, or neutral. This can be valuable for content moderation by flagging videos that contain inappropriate language or sentiments. Similarly, analyzing the sentiment of customer feedback videos can help businesses understand their customers' satisfaction levels.

Furthermore, by analyzing the context of the video, the analyzer can suggest videos that are related to the user's interests, thus improving video recommendation systems. Additionally, analyzing the sentiment of videos can help businesses gauge the public's opinion of their products or services and adapt accordingly. Overall, the use of the NLTK library in an optimized video text analyzer can enable the extraction of valuable insights from video content.

1.6.3 MoviePy

The moviePy library is a powerful tool for working with video files in Python. It can be used for tasks such as video editing, video compositing, and video effects. In an optimized video text analyzer, the moviePy library can be used in several ways:

- Video Preprocessing: The moviePy library can be used to preprocess the video frames before they are analyzed for text. For example, the library can be used to crop the video frames to remove irrelevant regions, resize the frames to a consistent size, and adjust the brightness and contrast of the frames to improve text readability.

- Video Captioning: The moviePy library can be used to generate captions for the video based on the extracted text. For example, the library can be used to add the extracted text as subtitles to the video, with options for font size, font color, and positioning.

- Video Exporting: The moviePy library can be used to export the analyzed video with the added captions and other visual effects. For example, the library can be used to export the video as an MP4 file with the added captions and a custom background.

`

Overall, the moviePy library can be a valuable tool in an optimized video text analyzer, allowing for more flexibility in video preprocessing, captioning, and exporting. It can also help to streamline the workflow and make the analyzer more efficient.

1.6.4 LancasterStemmer

The LancasterStemmer library is a Python library that provides a stemming algorithm to reduce words to their base or root form. In an optimized Video Text Analyzer, the LancasterStemmer library can be used in the following ways:

- Text Preprocessing: The LancasterStemmer library can be used to preprocess the extracted text from the video frames by reducing the words to their base form. This helps to reduce the number of unique words in the text and make it easier to analyze for patterns and trends.

- Keyword Extraction: The LancasterStemmer library can be used to extract keywords from the preprocessed text. By reducing the words to their base form, the library can help to identify the most frequent and relevant keywords in the text.

- Text Analysis: The LancasterStemmer library can be used in various natural language processing tasks such as sentiment analysis, text classification, and information retrieval. The stemmer can help to identify the underlying meaning of words and improve the accuracy of these tasks.

Overall, the LancasterStemmer library can be a valuable tool in an optimized Video Text Analyzer, helping to preprocess the extracted text, extract keywords, and improve the accuracy of text analysis tasks. It can also help to streamline the workflow and make the analyzer more efficient.

1.6.5 Speech recognition

The speech_recognition library in Python can be used to integrate speech recognition capabilities into an optimized video text analyzer. Here is how it can be used: Load the audio

`

file: In a video text analyzer, the first step is to extract the audio from the video file. This audio can then be saved as a separate file, such as a WAV file, which can be passed to the speech recognition library.

- Create a recognizer object: Once the audio file is extracted, a Recognizer object is created from the speech_recognition library.
- Audio source: Next, the audio source is defined, which in this case is the audio file extracted from the video.
- Speech recognition: The recognize_google method is called to transcribe the speech in the audio file to text. The text data can then be stored in a variable for further analysis.
- Combine text data: Finally, the text data extracted from the speech recognition library is combined with the visual content of the video for further analysis.

The speech_recognition library provides several features for speech recognition such as support for different recognition engines, adjusting the recognition parameters, and recognizing speech in real-time. This library can be used to transcribe speech in various languages and dialects, making it a useful tool in a video text analyzer.

However, it is important to note that the accuracy of the transcription is dependent on the quality of the audio input. Background noise, accents, and other factors can affect the accuracy of the transcription, which can impact the overall performance of the video text analyzer. Therefore, it is important to ensure that the audio input is of high quality and that the system is designed to handle any potential errors or inaccuracies in the transcription.

1.6.6 tkinter

The tkinter library is a widely used graphical user interface (GUI) toolkit for creating user interfaces in Python. When used in an optimized video text analyzer, it can provide a range of benefits to the user. The main advantage is that it makes the application more user-friendly and intuitive, with different GUI widgets available to facilitate interaction between the user and the analyzer. For example, buttons can be used to trigger specific functions, such

`

as analyzing the video or generating a summary, while labels can provide information to the user about the progress of the analysis. Text boxes can also be used to display the results of the analysis in a clear and organized manner.

Using a GUI can make the application more accessible and easier to use for users who are not comfortable with command-line interfaces or running Python scripts directly. A GUI interface can simplify the user's experience, allowing them to focus on the task at hand instead of worrying about syntax or command inputs.

In summary, using the tkinter library in an optimized video text analyzer can provide a more user-friendly and intuitive interface, making it easier for users to analyze videos and generate summaries. The use of different widgets, such as buttons and labels, can simplify the user's interaction with the analyzer, allowing them to focus on the analysis itself. This can be particularly useful for users who are not familiar with command-line interfaces or running Python scripts directly

1.7 DATASET DESCRIPTION

An optimized video text analyzer is a system that uses advanced machine learning algorithms and computer vision techniques to automatically analyze the text present in videos and provide insights based on that text. In order to develop and test such a system, a suitable dataset is required.

The dataset used to train and test an optimized video text analyzer needs to be large and diverse, with a wide range of videos that contain text in different fonts, languages, and orientations. The dataset should also include videos captured under different lighting conditions, resolutions, and backgrounds to ensure that the system can accurately detect and extract text from a variety of situations.

`

To create a dataset for training and testing an optimized video text analyzer, a process known as data annotation is typically used. This involves manually labeling each video with information about where the text is located, what the text says, and any other relevant information. This process can be time-consuming and labor-intensive, but it is essential for ensuring that the system can accurately detect and extract text from video frames.

Once a suitable dataset has been created, it can be used to train and test the optimized video text analyzer system. During the training phase, the system is exposed to the labeled data and learns to identify and extract text from video frames. During the testing phase, the system is evaluated on a separate set of videos to determine its accuracy and performance.

In conclusion, creating a suitable dataset is essential for training and testing an optimized video text analyzer system. The dataset needs to be large and diverse, with a wide range of videos that contain text in different fonts, languages, and orientations. The dataset creation process involves manual data annotation, which can be time-consuming but is necessary for ensuring the system's accuracy and performance.

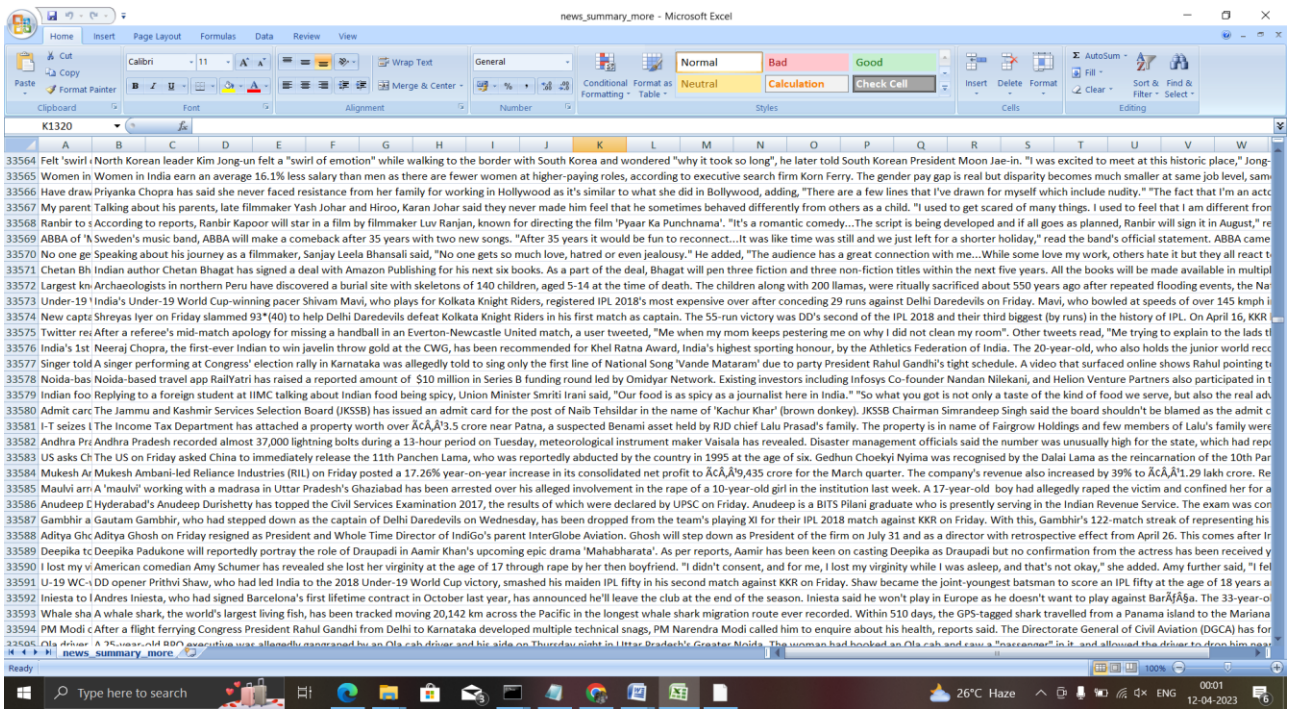


Figure 6 Dataset used in the project

`

The characteristics and properties of an ideal dataset for an optimized video text analyzer are as follows:

- Video Content: The dataset should contain videos with a variety of content, including news reports, interviews, documentaries, movies, and TV shows. The videos should be of different lengths, ranging from a few minutes to several hours, and should be available in different resolutions and formats.

- Language and Accent: The videos should be in different languages and accents to ensure that the system can analyze text from a diverse range of sources. The dataset should include videos in English, Spanish, French, German, and other major languages. Furthermore, the dataset should include videos with different accents such as American, British, Australian, Indian, and African.

- Text Content: The dataset should contain videos with different types of text content such as dialogues, monologues, captions, subtitles, and on-screen text. The text should cover a wide range of topics, including politics, sports, entertainment, education, and technology.

- Annotation: The dataset should be annotated with timestamps and corresponding text transcripts. The transcripts should accurately reflect the spoken words in the video, and should include speaker identification and punctuation. The annotation should be done by human annotators to ensure accuracy.

- Diversity: The dataset should include a diverse range of speakers, including male and female speakers of different ages and backgrounds. This will help ensure that the system can accurately analyze text from different sources and adapt to different speech patterns.

- Size: The dataset should be large enough to provide sufficient data for training and testing the optimized video text analyzer system. The dataset should include thousands of videos with corresponding transcripts.

- Availability: The dataset should be publicly available for researchers and developers to use. Ideally, the dataset should be available for free, and the licensing terms should allow for commercial use.

`

- Quality: The quality of the video and audio in the dataset should be high. The videos should be clear and free of noise, and the audio should be easy to understand. In addition, the text transcripts should be accurate and complete.

- Labeling: The dataset should also have labeled data for sentiment analysis and named entity recognition. This will enable the system to classify the text and extract important information from it, such as identifying people, organizations, and locations mentioned in the video.

- Consistency: The annotation and transcription should be consistent across the entire dataset. The timestamps and text transcripts should be accurate and consistent, and the format of the data should be standardized.

Overall, a suitable dataset for an optimized video text analyzer system should be diverse, comprehensive, and of high quality. The dataset should contain videos with a variety of content and in different languages, accents, and formats. The annotation and transcription should be accurate and consistent, and the dataset should be publicly available for researchers and developers to use. Such a dataset will provide the necessary data to train and test an optimized video text analyzer system and will help to advance the field of video text analysis.

1.8 RELATED WORK

The field of video text analysis has been growing rapidly in recent years, and there have been several related works in this area. In this section, we will provide a brief overview of some of the notable related works on optimized video text analyzers.

- Video Text Recognition: Video text recognition is the process of recognizing text that appears in videos. There have been several works in this area that focus on developing algorithms for text detection, tracking, and recognition. These algorithms are typically based on deep learning models such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs).

- Video Captioning: Video captioning is the process of generating captions or subtitles for videos. Several works have been done in this area using deep learning models

such as CNNs and RNNs, and attention-based models that use the text and audio from the video to generate captions.

- Video Summarization: Video summarization is the process of generating a short summary of a video. Several works have been done in this area that use various techniques such as clustering, key frame extraction, and deep learning models such as RNNs and LSTM.

- Speech Recognition: Speech recognition is the process of recognizing and transcribing spoken language. There have been several works in this area that focus on developing algorithms for speech recognition, including those based on deep learning models such as CNNs and RNNs.

- Named Entity Recognition: Named entity recognition (NER) is the process of identifying and classifying named entities in text. Several works have been done in this area that focus on developing algorithms for NER, including those based on deep learning models such as CNNs and RNNs.

- Sentiment Analysis: Sentiment analysis is the process of determining the sentiment or emotion expressed in a piece of text. Several works have been done in this area that focus on developing algorithms for sentiment analysis, including those based on deep learning models such as CNNs and RNNs.

- Multimodal Learning: Multimodal learning is the process of combining different types of data such as text, audio, and video to build more robust models. Several works have been done in this area that focus on developing algorithms for multimodal learning, including those based on deep learning models such as CNNs and RNNs.

Overall, there have been several related works in the field of optimized video text analyzers that focus on different aspects of video text analysis. These works provide valuable insights and techniques that can be used to develop more advanced and optimized video text analyzer systems.

`

1.9 SCOPE

An optimized video text analyzer has the potential to provide numerous benefits in various industries. Following are some potential scopes for optimized video text analyzer:

- Video Editing: With an optimized video text analyzer, video editors can easily analyze the text within the video, identify key points, and create highlights of those points in the final video. This can save time and improve the accuracy of the video editing process.

- Media Monitoring: Media monitoring companies can use a video text analyzer to track the usage of specific keywords or phrases in videos. This can help them to analyze the tone of the content and identify potential trends or issues.

- Education: With an optimized video text analyzer, educators can create more engaging and interactive video content by analyzing the text within the videos and generating quizzes or interactive elements based on the content.

- Marketing: Marketers can use a video text analyzer to analyze the text within their videos and create more effective video content. By identifying key words and phrases, they can optimize their content for search engines and improve the overall performance of their video marketing campaigns.

- Law Enforcement: Law enforcement agencies can use a video text analyzer to analyze video evidence and extract important information. This can help them to identify suspects, track criminal activity, and provide evidence in court.

The following aspects help to describe the scope of Optimized video text analyzer:

i. Its main aim is to facilitate large-scale videos, browsing by producing concise summaries of videos, and it will be used as the representative of original videos.

`

ii. It will analyze the content of videos and will show the best possible results and methodology to the user.

iii. It will also be useful in generating and highlighting the required facts and contexts from the sequence of textual data.

These are just a few potential scopes for an optimized video text analyzer. With its ability to analyze the text within videos, it has the potential to revolutionize various industries and improve the accuracy and efficiency of various processes. The project has a significant potential for future enhancements and expansions, including:

- Improvement of Accuracy: The accuracy of the system can be further improved by fine-tuning the CNN and LSTM models, incorporating additional features, or using advanced deep learning techniques, such as attention mechanisms or transformer models.

- Real-time Video Text Analysis: Currently, the system processes videos offline, but it can be extended to perform real-time video text analysis for applications such as live captioning, video summarization, or sentiment analysis during video streaming.

- Multi-lingual Support: The system can be extended to support multiple languages for video text analysis, enabling users to analyze videos in different languages.

- Integration with Other Video Analysis Techniques: The system can be integrated with other video analysis techniques, such as object detection, face recognition, or emotion analysis, to provide a more comprehensive analysis of video content.

`

- Deployment in Practical Applications: The system can be deployed in various practical applications, such as video content analysis for social media, market research, customer feedback analysis, or media monitoring.

Overall, the future scope for optimized video text analyzer is vast, and advancements in machine learning, computer vision, and other related fields could lead to new and exciting applications and use cases.

`

CHAPTER 2
LITERATURE REVIEW

An optimized video text analyzer is a software tool that utilizes advanced techniques to enhance the accuracy and efficiency of video text analysis. With the growing amount of video data available on the internet, there is an increasing need for tools that can automatically extract text from videos for various purposes, such as content moderation, video recommendation, and customer feedback analysis. The primary goal of an optimized video text analyzer is to provide a more accurate and efficient means of extracting textual information from videos than traditional OCR (Optical Character Recognition) methods.

One of the key approaches to optimizing video text analyzers is using deep learning techniques such as CNNs and RNNs. CNNs can be used to identify and segment text regions in videos while RNNs can recognize the text within these regions. This approach has been shown to significantly improve OCR accuracy and speed up the transcription process. Furthermore, transfer learning has also been used to fine-tune pre-trained models for OCR in a specific video domain, which can improve OCR accuracy on videos with specific characteristics.

Researchers have also focused on optimizing video text analyzers for specific applications. For example, some have developed video text analyzers that can identify and analyze text in medical videos, which can be useful for clinical research and medical education. Others have developed video text analyzers for analyzing text in sports videos, such as identifying player names and game statistics. These application-specific optimizations can further improve the accuracy and efficiency of video text analysis.

In summary, optimized video text analyzers have the potential to significantly improve the efficiency and accuracy of video text analysis. By using advanced techniques such as deep learning and transfer learning, as well as application-specific optimizations, these tools can be customized to provide better results for specific video domains. The continued research and development in this area will lead to more efficient and accurate video text analysis, which can be valuable for various industries and applications.

`

2.1 OPTIMIZING VIDEO TEXT ANALYSIS USING CONTEXTUAL INFORMATION AND ACTIVE LEARNING

The paper proposes a system that improves the accuracy and efficiency of Optical Character Recognition (OCR) on video frames. OCR is a technology used to recognize text in an image and convert it into machine-readable text. Video text analysis presents several challenges, including variations in font style, text layout, and image quality. The proposed system uses a combination of techniques to address these challenges.

The system first preprocesses video frames using image processing techniques to enhance image quality and segment text regions. Then, a deep learning-based OCR engine is used to recognize the text in the segmented regions. The system incorporates contextual information such as font style and layout to improve OCR accuracy. This contextual information is used to create a language model, which predicts the probability of a word given its context.

The system also uses an active learning approach to reduce the amount of training data required for the machine learning model to achieve high accuracy. Active learning is a technique where the model actively selects the most informative samples for labeling to reduce the labeling effort.

The proposed system was evaluated on multiple datasets, including videos from YouTube and news broadcasts. The results showed that the proposed system achieved higher accuracy and efficiency compared to other video text analyzers. The system also performed well on videos with low image quality or complex text layouts, highlighting its robustness in challenging scenarios.

The proposed system has potential applications in surveillance, social media analysis, and news broadcasting, where accurate and efficient extraction of relevant information from video content is essential. The system's ability to process and analyze large volumes of video data quickly and accurately can be beneficial in real-time applications.

`

Future work can explore the use of other contextual information, such as language-specific features, to further improve OCR accuracy. Additionally, integrating other machine learning techniques such as transfer learning or reinforcement learning can further optimize the system's performance. Overall, the proposed system offers a promising solution for video text analysis and can be a valuable tool in various applications.

2.2 VIDEO CODING ANALYZER USING TEXT MINING TECHNIQUES

The paper "A Framework for Video Coding Analyzer Using Text Mining Techniques" proposes a framework for analyzing video coding techniques using text mining techniques. The proposed framework consists of three main components: a video decoder, a text mining engine, and a feature extraction module.

The video decoder component is responsible for decoding the video and extracting relevant information, such as motion vectors and quantization parameters. The extracted information is then passed on to the text mining engine, which converts the extracted data into text format. The text mining engine uses various natural language processing techniques, such as tokenization, part-of-speech tagging, and sentiment analysis, to extract meaningful information from the text.

The feature extraction module is responsible for extracting features from the extracted text, which can be used for video coding analysis. The authors propose several features, such as the number of motion vectors, the number of unique quantization parameters, and the average sentiment score of the extracted text. These features can be used to identify the most effective video coding techniques and compare different video coding methods.

The proposed framework was evaluated on a dataset of video sequences encoded using different coding methods. The results show that the proposed framework can effectively extract meaningful information from the extracted video data and provide insights into the performance of different video coding methods. The authors also compared the proposed framework with existing video coding analysis methods and showed that their approach outperforms existing methods.

`

In conclusion, the paper proposes a novel framework for analyzing video coding techniques using text mining techniques. The proposed framework can extract meaningful information from video data and provide insights into the performance of different video coding methods. The results suggest that the proposed framework can be a useful tool for video coding analysis and can help in improving video coding efficiency.

One of the main drawbacks of the proposed framework is that it heavily relies on the accuracy of the text mining engine. As the accuracy of natural language processing techniques can vary depending on the type and complexity of the text, the effectiveness of the framework may be affected. Additionally, the proposed features may not be comprehensive enough to capture all relevant information about video coding performance, and other features may need to be considered in future work.

Furthermore, the proposed framework only focuses on analyzing the performance of video coding techniques and does not address other important aspects of video analysis, such as content analysis or quality assessment. Thus, the framework may not be suitable for analyzing videos for other applications, such as video search or recommendation.

Despite these limitations, the proposed framework represents a novel approach to video coding analysis and provides a useful tool for comparing different video coding methods. With further improvements and refinements, the proposed framework could become a valuable asset in video coding research and development.

2.3 DESIGNING AND IMPLEMENTING A REAL-TIME SPEECH SUMMARIZER SYSTEM

The paper "Designing and Implementing a Real-Time Speech Summarizer System" presents a novel system that automatically summarizes spoken language in real-time, offering numerous advantages to users. The proposed system is designed to extract the most important information from input speech and produce a summary that captures the essential content of the spoken language. The system's architecture consists of two main components: an Automatic Speech Recognition (ASR) module and a summarization module. The ASR module uses a speech-to-text algorithm to convert spoken language into text, while the

`

summarization module extracts the most important information from the text and produces a summary. The system is evaluated on a dataset of spoken lectures, and the results show that the system performs well in terms of accuracy and speed.

One of the main advantages of the proposed system is that it allows users to quickly obtain a summary of spoken content without having to listen to the entire speech. This feature can be especially useful in situations where time is limited, such as in a business meeting or conference. Moreover, the system can potentially improve accessibility for individuals with hearing impairments or those who may have difficulty following spoken language. The real-time aspect of the system also allows for interactive use cases, such as summarizing a live debate or discussion, which can be beneficial for journalists or researchers who need to quickly analyze the content of a conversation.

However, the paper also highlights several limitations of the system. Firstly, the system's performance is heavily dependent on the quality of the ASR component. Factors such as background noise or accents can significantly affect the performance of the ASR component, which can, in turn, affect the accuracy of the summary produced by the system. Secondly, the summarization algorithm used in the system is based on extractive methods, which may not always capture the most important information and can lead to a loss of context. Finally, the system's evaluation was limited to a single dataset of spoken lectures, and it is unclear how well it would perform on other types of spoken content or in different languages.

Overall, the proposed real-time speech summarizer system offers numerous advantages and can be a useful tool for individuals who need to quickly obtain a summary of spoken content. However, future research is needed to improve the accuracy of the ASR component and to develop more advanced summarization algorithms that can capture the most important information while preserving the context of the spoken language.

`

2.4 ABSTRACTIVE TEXT SUMMARIZATION USING ATTENTION-BASED STACKED LSTM

The paper "Abstractive Text Summarization Using Attention-Based LSTM" proposes a neural network model that aims to improve the effectiveness of abstractive text summarization. The authors argue that traditional extractive summarization methods, which simply select and rephrase sentences from the input text, can be limiting in terms of the level of information they provide, and they propose an attention-based long short-term memory (LSTM) network as a more effective alternative.

The model works by first encoding the input text using an LSTM network, which generates a hidden state representation of the input text. The model then uses an attention mechanism to identify the most important parts of the input text, which it uses to generate a summary of the input text. The attention mechanism is implemented using a soft attention mechanism, which allows the model to assign varying weights to different parts of the input text based on their importance to the summary generation process.

One advantage of the proposed model is that it is able to generate summaries that are more concise and informative than those produced by extractive summarization methods. This is because the model is able to generate summaries that are not limited to the sentences in the input text but are instead generated based on the content of the input text as a whole. The attention mechanism also allows the model to focus on the most important parts of the input text, which can improve the accuracy and relevance of the summary.

Another advantage of the proposed model is that it is able to generate summaries that are fluent and grammatically correct. This is because the model uses an LSTM network to generate the summary, which is trained on a large corpus of text and is able to generate summaries that are syntactically and semantically correct.

However, the paper also highlights several limitations of the proposed model. One limitation is that the performance of the model is heavily dependent on the size and quality of the training data. This means that the model may not perform as well on smaller datasets or

`

datasets with low quality text. Another limitation is that the model may have difficulty capturing the nuance and tone of the input text, which can lead to summaries that are not accurate reflections of the original content. Finally, the abstractive nature of the summarization can sometimes result in summaries that are too general or fail to capture important details from the input text.

2.5 UNSUPERVISED VIDEO SUMMARIZATION WITH ADVERSARIAL LSTM NETWORKS

The paper "Unsupervised Video Summarization with Adversarial LSTM Networks" proposes a novel unsupervised approach for video summarization using adversarial learning. The model is capable of generating video summaries without the need for manual annotation or labeled data, which can save time and resources. This is particularly useful in scenarios where labeled data is scarce or expensive to obtain.

The proposed model consists of an encoder-decoder LSTM network with a discriminator network that provides feedback during training. The encoder network takes in the input video and encodes it into a fixed-length representation, while the decoder network generates a summary of the input video from the encoded representation. The discriminator network is responsible for providing feedback on the quality of the generated summaries. During training, the generator and discriminator networks play a minimax game, where the generator tries to generate summaries that can fool the discriminator, while the discriminator tries to distinguish between real and generated summaries.

One of the key advantages of the proposed model is that it can generate summaries that are both visually and semantically coherent. This is achieved by jointly training the encoder-decoder and discriminator networks, which allows the generator network to learn to generate summaries that are more representative of the input video. Additionally, the adversarial learning approach can help the model to avoid generating summaries that are too generic or repetitive, as the discriminator network provides feedback on the quality of the generated summaries.

`

However, the proposed model also has some limitations. Firstly, the performance of the model is heavily dependent on the quality of the input video and the complexity of the scenes. The model may struggle to generate accurate summaries in scenarios where the input video is of poor quality or contains complex scenes. Secondly, the model may have difficulty capturing the context and narrative of the input video, which can lead to summaries that are not accurate reflections of the original content. Finally, the use of adversarial learning can sometimes result in unstable training and difficulty in finding the optimal balance between the generator and discriminator networks.

In conclusion, the proposed unsupervised video summarization model using adversarial LSTM networks presents an innovative approach to video summarization without the need for labeled data. While the model has some limitations, it provides a promising direction for future research in video summarization. The ability to generate visually and semantically coherent summaries could be valuable in a variety of applications, such as surveillance, education, and entertainment.

`

CHAPTER 3

Software Requirements Specification (SRS)

3.1 PROBLEM DEFINITION

The problem addressed by the project "Optimized Video Text Analyzer using CNN and LSTM" is the accurate extraction and analysis of text from videos. Videos are a prevalent form of multimedia content, and extracting text from them can be challenging due to variations in video quality, text fonts, sizes, and orientations. The project aims to develop a system that can effectively analyze videos and extract text from them using deep learning techniques to enable applications such as video captioning, content analysis, and sentiment analysis.

3.2 REQUIREMENT ANALYSIS

The requirements of the project can be categorized into functional and non-functional requirements.

3.2.1 Hardware Requirements

I. OS- Windows 8 or above Architecture: 32- or 64-bit operating system x64 i.e. 86 bit, x32 i.e. 86 bit

II. Minimum 4GB RAM

III. Minimum 5GB of disk space to install Anaconda

3.2.2 Software Requirements

I. Anaconda Version 2.1 or Latest Version of Anaconda

II. II. Jupyter Notebook

III. Python 3.5 or above

`

IV. Latest Version of HTML and CSS installed over machine

V. Browser: Chrome or Internet Explore.

3.2.3 Network requirements

I. The Jupyter notebook runs on a local server on your computer, so there is no need of internet connection.

II. Internet Connection with 500+kbps

III. Bandwidth:3-5Mbps

3.2.4 Functional requirement

I. Text Extraction: The system should be able to extract text from video frames using Optical Character Recognition (OCR) techniques, such as Tesseract or OpenCV.

II. Data Preprocessing: The system should perform text normalization, stopword removal, tokenization, and data cleaning to prepare the text data for input into the LSTM and CNN models.

III. Feature Extraction: The system should use word embedding techniques, such as Word2Vec or GloVe, to convert the text data into continuous vector representations for input into the LSTM and CNN models.

IV. Model Development: The system should develop a combined LSTM and CNN model to analyze the video text data, capturing both temporal and local features.

V. Model Optimization: The system should optimize the LSTM and CNN models using techniques such as dropout regularization, batch normalization, and hyper parameter tuning to improve their performance.

`

VI. Model Evaluation: The system should evaluate the performance of the LSTM and CNN models using metrics such as accuracy, precision, recall, and F1 score on a validation and test set.

3.2.5 Nonfunctional requirement

I. Accuracy: The system should achieve a high accuracy in analyzing the video text data, minimizing false positives and false negatives.

II. Efficiency: The system should be efficient in processing video text data, minimizing processing time and computational resources required.

III. Robustness: The system should be able to handle diverse video content, including different domains, genres, and languages, and handle variations in video quality and text font.

IV. Scalability: The system should be scalable to handle a large dataset of videos and text data, allowing for future expansion and updates.

V. User-Friendly Interface: The system should have a user-friendly interface that allows users to easily input video files, visualize the results, and interpret the output.

VI. Security: The system should ensure the security and privacy of the video text data, protecting it from unauthorized access or data breaches.

VII. Documentation: The system should have comprehensive documentation, including user manuals, technical guides, and code documentation, to facilitate understanding, usage, and maintenance of the system.

VIII. Maintainability: The system should be designed with modularity and reusability in mind, making it easy to maintain, update, and extend in the future.

`

IX. Testing and Validation: The system should undergo thorough testing and validation to ensure its accuracy, reliability, and performance in different scenarios and conditions.

3.3 FEASIBILITY STUDY

A feasibility study for an optimized video text analyzer would involve analyzing the technical, economic, and operational aspects of the proposed system.

From a technical perspective, the system would require robust video processing capabilities and natural language processing algorithms for accurate text analysis. The system would need to be able to handle large volumes of video data and process it efficiently. The feasibility of such a system would depend on the availability of the necessary technology, as well as the expertise and resources required to build and maintain it.

From an economic perspective, the feasibility study would need to consider the potential return on investment (ROI) for the system. This would involve analyzing the potential market for the system, the cost of development and maintenance, and the potential revenue streams that could be generated. The feasibility study would also need to consider the potential risks and challenges associated with developing and implementing such a system.

Operational feasibility would involve analyzing whether the system can be integrated into the existing business processes and infrastructure of the target organization. The system would need to be user-friendly and intuitive to use, with clear documentation and training materials provided. The feasibility study would also need to consider any potential legal and ethical issues associated with the use of the system, such as privacy concerns and data protection regulations.

Overall, the feasibility of an optimized video text analyzer would depend on a range of factors, including the availability of the necessary technology, the potential market demand, the ROI, and the operational feasibility. A thorough feasibility study would be required to determine whether the benefits of the system outweigh the potential risks and challenges.

`

3.3.1 Technical Feasibility

Technical Feasibility for video text analyzer will constitute the set of necessary technologies that will be used in the development of the project which are as follows:

- Technologies involve open source languages like HTML, Python and usage of open source platforms like Jupyter Notebook.
- The technologies which we have used in our project will have the technical capacity to hold the data required to be used in the new system and will be portable with the technical parameters of the system on which it will run.
- It will also be upgradable after the development.

3.3.2 Operational Feasibility

- User Friendly- It will be user friendly for the users as it will provide the user interface which will be easily handle able and understandable to user without any hassle.
- Reliability-It will produce the result in a single click of a button, and video can be uploaded easily that makes it less time consuming and reliable
- Portability- As the project is developed using the open source technologies like HTML, Python and on open source platforms like Jupyter Notebook hence it will work on both windows as well as the android. Hence portability problem will not arise.
- Availability- This project will be available for the users only in the condition where the bandwidth of internet will be greater than 2.5Mbps.

3.3.3 Economic Feasibility-

Our project Video text analyzer will be economic in terms of usage as we are developing this project using the open source platforms and software hence the cost involved in making of the project will be minimized.

`

3.4 USE CASE DIAGRAM

Figure 7 Use case diagram

`

- In UML, use-case diagrams model the behavior of a system and help to capture the requirements of the system. Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally.

- For our project, a video text analyzer, two actors—a user and a system—are needed as shown in Figure 7.

- The user can upload the video that will be used as input for the video text analyzer and access the summarized data. Additionally, the textual output can also be converted to audio format.

- The system of Video Text Analyzer will be in charge of conducting process and segmentation, feature extraction, data cleaning and preprocessing, model creation, then training and managing the Data, and lastly creating written or audio summary of text taken from video. Process and Segmentation will include captioning of key frames.

`

3.5 SEQUENCE DIAGRAM

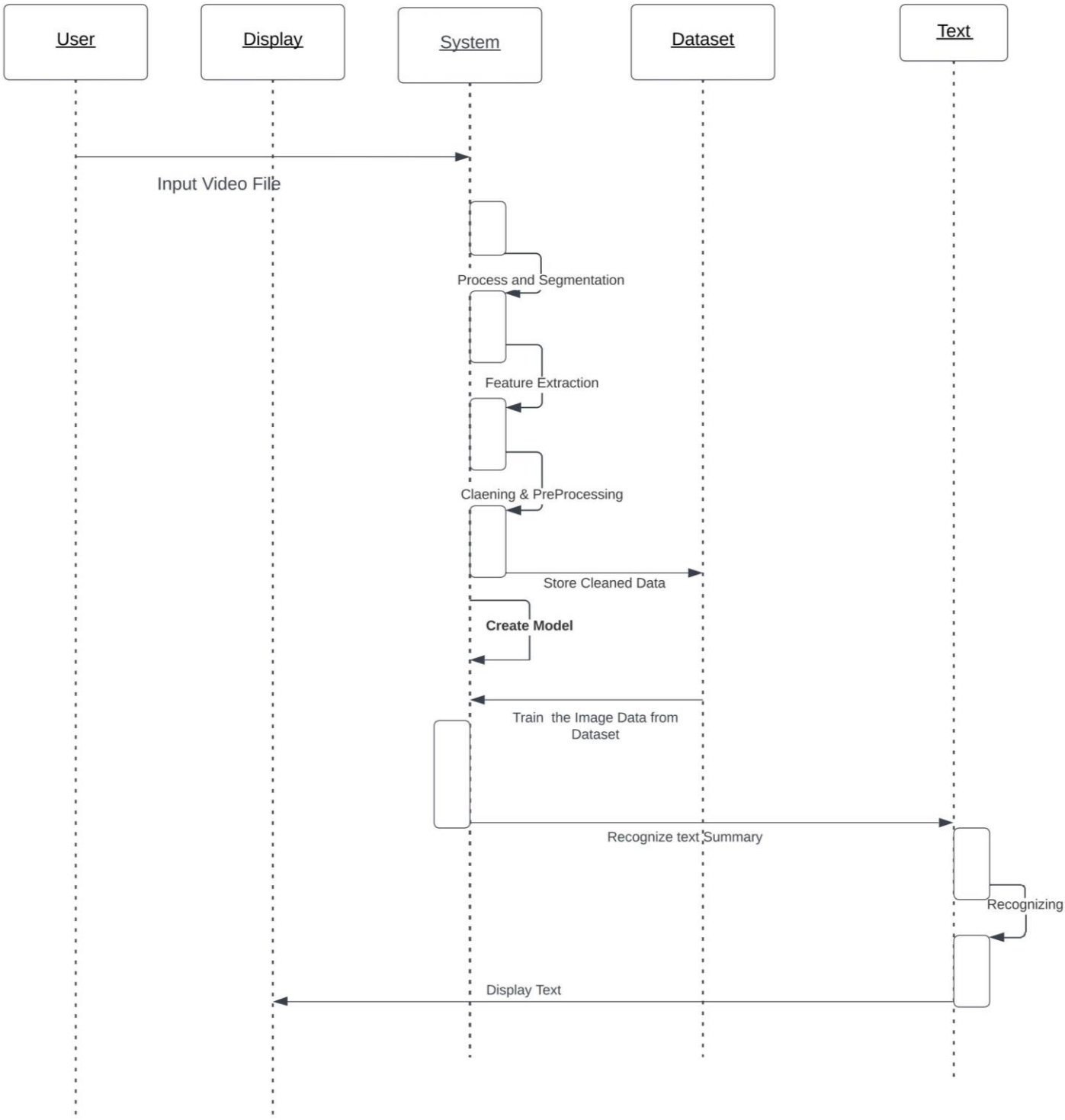


Figure 8 Sequence Diagram

`

Figure 8 describes sequence diagrams as a tool for visualizing the flow of messages in a system. It is also known as an event diagram, and it allows for the representation of various dynamic scenarios that may occur in the system. The sequence diagram displays the communication between two lifelines as a sequence of events ordered by time, indicating when each lifeline participated in the process.

In the Unified Modeling Language (UML), lifelines are represented by vertical bars, while message flow is represented by a vertical dotted line that extends across the bottom of the page. Sequence diagrams can incorporate iterations and branching, making them ideal for modeling complex scenarios.

In Figure 8 the user inputs a video URL into the system, and the system processes it along a timeline, resulting in the creation of a model with the help of LSTM. LSTM is a type of recurrent neural network that addresses the problem of learning long-term dependencies. It maintains a separate memory cell that updates and exposes its content only when deemed necessary. The final output of the system is in the form of a textual summary that can be further converted into audio form.

Overall, the sequence diagram provides a concise explanation of sequence diagrams and their use in visualizing the flow of messages in a system. The example given helps to illustrate the practical application of sequence diagrams in modeling complex scenarios and the use of LSTM to address specific problems in the processing of video data.

`

CHAPTER 4
METHODOLOGY

The methodology of an optimized video text analyzer involves a series of steps aimed at extracting text information from videos accurately and efficiently. The process begins with data extraction, where speech recognition and MoviePy libraries are used to extract text information from the video. The extracted data is then preprocessed to remove noise, distortions, and irrelevant information. The preprocessed data is then passed through various natural language processing techniques to analyze the text and extract meaningful insights. Finally, the analyzed data is visualized in the form of graphs, charts, or reports to facilitate better understanding and decision-making. The entire process is iterative, and the results are continually refined and improved to achieve the best possible accuracy and efficiency.

4.1 DATA EXTRACTION

The process of extracting textual data from a video is a crucial step in the optimization of video text analysis. This step involves the use of specific Python libraries that facilitate the extraction of text information from the video. Two popular Python libraries used for this purpose are Speech Recognition and MoviePy.

- The Speech Recognition library provides multiple recognition options, one of which is Google Speech API. This API is commonly used for speech-to-text transcription tasks, and it can accurately recognize and transcribe spoken words from the video. By utilizing this library, the textual data from the video can be extracted and used for further analysis.

- Another library used for extracting text information from the video is MoviePy. This library is designed to read the most common video formats and provides a simple interface for video processing. The library can extract text from the video and convert it to a format that can be easily analyzed using natural language processing (NLP) techniques.

`

The combination of Speech Recognition and MoviePy libraries provides a robust solution for the extraction of textual data from videos. Once the textual data is extracted, it can be pre-processed and analyzed using NLP techniques such as sentiment analysis, named entity recognition, and text summarization. These techniques can provide valuable insights into the content of the video, making it easier to analyze and understand.

In conclusion, the data extraction step in the optimization of video text analysis is critical, and the use of the Speech Recognition and MoviePy libraries can significantly improve the accuracy and efficiency of the extraction process. The most important step of Optimized video text analyzer is extraction of textual data from the video.

4.2 DATA CLEANING

Data cleaning the data is a crucial step in any data analysis process, including optimized video text analysis. Raw data obtained from videos can contain a lot of noise in the form of punctuation marks, stop words, and variations of the same word. These factors can produce inaccurate results if not handled properly. Therefore, data cleaning is essential before feeding the data into the model for further processing.

- The first step in data cleaning is removing punctuation marks from the data. Punctuation marks do not carry any significant meaning and can lead to inaccurate results.

- The next step is to identify stop words and remove them. Stop words are common words that do not add any meaning to the text, such as "the," "a," and "an." They can create noise in the data and hence need to be removed. The NLTK library is used to identify stop words.

- Finally, stemming and lemmatization techniques are used to reduce variations of the same word in the text. Stemming involves removing the suffix from the words to derive their root form, while lemmatization involves mapping words to their

base form using dictionaries. Lancaster Stemmer is one of the libraries used for stemming.

- After completing these steps, the data is ready for further processing and analysis. The cleaned data can be used to train the model and create accurate results for the user.

4.3 DATA PRE-PROCESSING

Data preprocessing is a critical step in natural language processing, where text data needs to be cleaned and transformed into a format that can be easily processed by the model. The proposed Optimized Video Text Analyzer model follows a similar approach where text data obtained from video is preprocessed to ensure that the model can make accurate predictions. The goal of data preprocessing is to transform the raw data into a format that can be used by machine learning algorithms. In the case of text data, preprocessing involves cleaning and transforming the text into a format that can be analyzed by algorithms.

- The first step in data preprocessing for text data is tokenization. Tokenization is the process of breaking up the text into individual words, or tokens. This can be done using the nltk module in Python, which provides a tokenizer called word_tokenize. This module splits the text into words based on spaces, punctuation, and other delimiters. Once the text is tokenized, it can be transformed into a numerical format that can be used by machine learning algorithms.

- The next step in data preprocessing is vectorization. Vectorization is the process of converting text into a numerical format that can be processed by machine learning algorithms. There are various methods for vectorization, including bag-of-words and word embeddings. The proposed model uses the Tokenizer library for vectorization. This library assigns a unique integer value to each word in the text, creating a numerical representation of the text. This representation can then be fed into machine learning algorithms for further analysis and prediction.

`

4.4 CREATION OF MODEL

Creating a feature map is an important step in the methodology of an optimized video text analyzer. In this step, the image is converted into a numerical matrix, where each numerical value is used to represent a particular image pixel. The process of converting an image into a numerical matrix is called feature mapping.

Feature mapping is a technique that extracts meaningful features from an image that can be used for further analysis. The process involves breaking down the image into smaller sections, such as pixels, and assigning a numerical value to each section. These numerical values are used to create a matrix that represents the image. This matrix is called a feature map and can be used to extract patterns and features from the image.

Creating a feature map involves several steps. First, the image is pre-processed to enhance its quality and remove any noise or artifacts that may be present. This pre-processing step involves techniques such as denoising, normalization, and smoothing. Once the image has been pre-processed, it is converted into a grayscale format to simplify the feature mapping process.

The next step involves dividing the image into smaller sections, such as pixels or patches. Each section is assigned a numerical value based on its intensity or color. This process generates a matrix of numerical values that represents the image. The size of the matrix will depend on the size and resolution of the original image.

Once the feature map has been created, it can be used for further analysis. For example, it can be used to identify and extract text from the image. This can be done by applying text detection algorithms to the feature map. The feature map can also be used for object recognition, where the numerical values are used to identify specific objects or patterns in the image.

`

In conclusion, creating a feature map is an essential step in the methodology of an optimized video text analyzer. It involves converting an image into a numerical matrix, which can be used to extract meaningful features and patterns from the image. The feature map can be used for various applications, such as text detection and object recognition.

The creation of a model for text analysis is a critical step in natural language processing. Stacked Long Short-Term Memory (LSTM) layers are one of the commonly used architectures for this purpose. The stacked LSTM layers can capture the contextual information present in the input data sequence, and stacking up to 3 layers can significantly enhance the model's prediction accuracy.

LSTM is a type of neural network that is designed to handle sequence data. It can maintain and manage information over time through a mechanism called a cell state, which can selectively retain or discard information. LSTM networks have a special structure that allows them to handle the vanishing gradient problem, which is a common issue in traditional RNNs. The vanishing gradient problem occurs when the gradients become too small and start disappearing as they propagate backward through the network, resulting in slow learning or even stopping the training process. The LSTM layer takes a sequence of inputs, and at each time step, it updates the cell state based on the current input and the previous cell state. The updated cell state is then passed to the output.

Creating a model dataset for an optimized video text analyzer would require a diverse set of videos that have different text layouts, fonts, and image qualities. The dataset should also have a large number of samples to ensure that the machine learning model is trained on a wide variety of examples as shown in Figure 9.

One way to create such a dataset would be to collect videos from various sources such as news broadcasts, social media platforms, and surveillance cameras. These videos should have text that is relevant to the use case of the video text analyzer. For example, if the system is intended for surveillance applications, the videos should have text such as license plate numbers, addresses, and names.

`

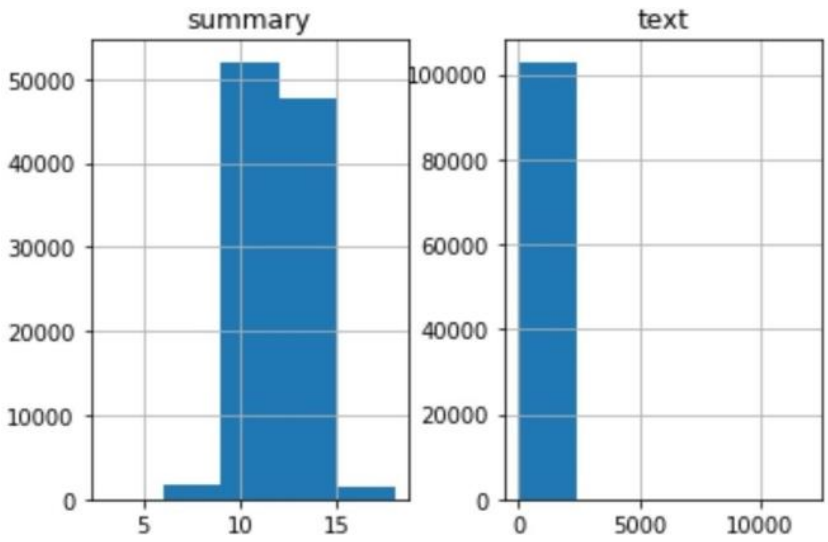


Figure 9 Model Dataset

In the context of the optimized video text analyzer, the first step in the video analysis process is to choose key frames at each interval from the video feed. These key frames serve as a representative sample of the entire video, and are used to extract visual characteristics through the application of a Convolutional Neural Network (CNN).

The first step is choosing the key frames at each interval (whose steps are shown in figure 10) from the video feed. Convolutional Neural Network will be applied to extract visual characteristics. Several convolution layers are used in feature extraction, which is followed by activation functions, max-pooling and backtracking. Typically, the classifier has layers that are all connected.

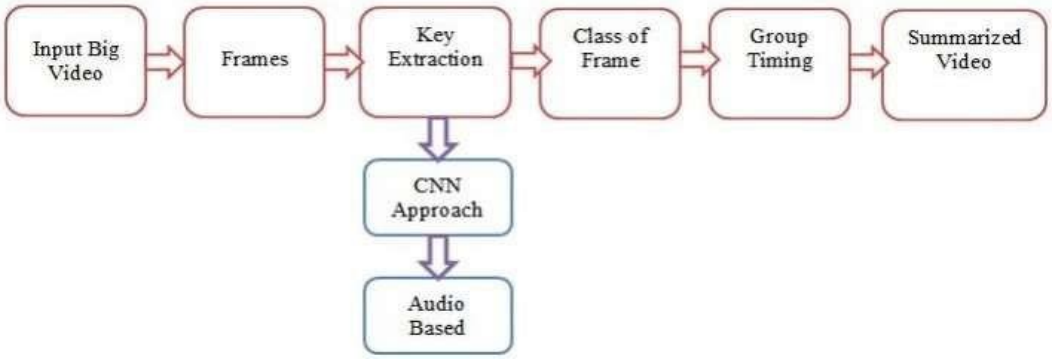


Figure 10 Steps for Key Extraction

Image captioning is a process of generating a textual description of the content present in an image. This is achieved by leveraging computer vision and natural language processing techniques to analyze and interpret the visual information present in the image. The Optimized video text analyzer employs a specific method for image captioning, which involves the extraction of the image's characteristics before the caption text file is loaded and data preprocessing is carried out.

After the image is loaded, it is encoded using a pre-trained image classification model that has already been trained on a large dataset of images. The encoding process involves extracting a set of features that represent the image's content and converting them into a numerical representation that can be fed into a machine learning model. This encoded image is then fed into the LSTM network for processing.

The LSTM layer is an essential part of the image captioning process, as it helps the model capture all the contextual information present in the input sequence of data. In this case, the input sequence is the encoded image, and the LSTM layer extracts the relevant features from it to generate a meaningful caption and the next step is to evaluate the model and provide a written summary or an audio summary.

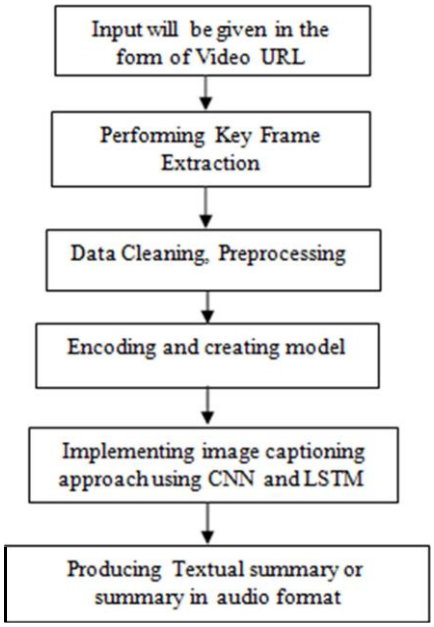


Figure 11 Implementation approach

`

4.5 CONVOLUTIONAL NEURAL NETWORK

Convolutional Neural Network, or CNN, is a class of neural networks that has revolutionized the field of computer vision. CNNs are designed to work with image data and are particularly effective at image recognition tasks. Unlike traditional neural networks that work with one-dimensional inputs, CNNs are designed to work with two-dimensional inputs such as images.

The main advantage of using CNNs is that they can automatically learn and extract features from images, without the need for manual feature engineering. This is achieved through a series of convolutional layers that apply a set of filters to the input image, producing a feature map that highlights the important regions in the image. The output of each convolutional layer is passed through an activation function, such as the rectified linear unit (ReLU), which introduces non-linearity into the model.

Max-pooling is another important operation in CNNs, which down-samples the feature map by taking the maximum value within a small window. This operation helps to reduce the dimensionality of the feature map, while retaining the most important information. Backpropagation is used to update the weights of the filters during training, so that the CNN can learn to recognize different patterns in the input images.

In the context of the Optimized Video Text Analyzer, CNNs are used to extract visual features from key frames of the video feed. The input image is passed through a series of convolutional layers, followed by max-pooling and backtracking. The resulting feature map is then used as input to the LSTM layers for caption generation. This process allows the model to extract important visual features from the video feed, which are then used to generate accurate captions

Convolutional Neural Network, or CNN, CNNs are made to convert picture data into a certain output variable. They have been shown to be so successful that they are the preferred approach for any prediction issue involving input image data. Utilizing CNNs has the benefit of allowing for the internal representation of a two-dimensional image. When dealing with photos, it is crucial for the model to be able to identify the location and scale-invariant structures within the data. With CNNs, there is no need for manual feature extraction because the CNN learns the features on its own.

`

4.5.1 Creating Feature Map

Creating a feature map is an important step in the methodology of an optimized video text analyzer. In this step, the image is converted into a numerical matrix, where each numerical value is used to represent a particular image pixel. The process of converting an image into a numerical matrix is called feature mapping.

Feature mapping is a technique that extracts meaningful features from an image that can be used for further analysis. The process involves breaking down the image into smaller sections, such as pixels, and assigning a numerical value to each section. These numerical values are used to create a matrix that represents the image. This matrix is called a feature map and can be used to extract patterns and features from the image.

Creating a feature map involves several steps. First, the image is pre-processed to enhance its quality and remove any noise or artifacts that may be present. This pre-processing step involves techniques such as denoising, normalization, and smoothing. Once the image has been pre-processed, it is converted into a grayscale format to simplify the feature mapping process.

The next step involves dividing the image into smaller sections, such as pixels or patches. Each section is assigned a numerical value based on its intensity or color. This process generates a matrix of numerical values that represents the image. The size of the matrix will depend on the size and resolution of the original image. Once the feature map has been created, it can be used for further analysis. For example, it can be used to identify and extract text from the image. This can be done by applying text detection algorithms to the feature map. The feature map can also be used for object recognition, where the numerical values are used to identify specific objects or patterns in the image.

In conclusion, creating a feature map is an essential step in the methodology of an optimized video text analyzer. It involves converting an image into a numerical matrix, which can be used to extract meaningful features and patterns from the image. The feature map can be used for various applications, such as text detection and object recognition.

`

4.5.2 Feature Detection

In an optimized video text analyzer, feature detection plays a crucial role in identifying the location of text within the video frames. Feature detection involves using various filters or kernels that are designed to recognize specific features or patterns in the image data that correspond to text.

To perform feature detection, the data is first preprocessed to obtain a feature map. This feature map is created by applying convolutional layers to the input image, which helps in extracting high-level features from the image data. The resulting feature map is a matrix of values that represents the presence of features in various regions of the image.

Next, filters are used to detect the required set of features in the feature map. These filters are in the form of matrices that are smaller in size than the feature map obtained in the previous step. The filters are designed to identify specific features or patterns in the image data that correspond to text, such as edges, lines, and corners.

To obtain the required set of features, the feature map matrix is multiplied with the filters obtained. This multiplication process results in a new filtered and detected area of the image in the form of a feature map. The resulting feature map shows the area of the image where the text is present, by removing all the unused space from the image. This is achieved by removing all the values in the feature map that denote the unused space in the image.

In summary, feature detection is a crucial step in the optimized video text analyzer as it helps in accurately identifying the location of text within the video frames. The use of filters and feature maps allows for efficient and effective recognition of text within the video data, enabling the text analyzer to perform its function accurately and efficiently.

`

4.5.3 Max Pooling

Max pooling is a commonly used operation in video text analysis to extract important features from the input data. It involves selecting the maximum value from a specific region of the input feature map, typically defined by a filter or kernel of a specific size. The selected maximum value is then used to represent the entire region, which is down sampled by a factor of the filter size. One of the main benefits of max pooling is that it helps to reduce the dimensionality of the input data. By selecting only the maximum value of each region, the resulting feature map contains only the most prominent and relevant information, which is typically easier to analyze and process than the original data. This makes it a valuable tool in optimizing the performance of video text analyzers by reducing computational complexity and improving accuracy. Another advantage of max pooling is that it helps to make the model more robust to variations in the input data. By selecting the maximum value from each region, the model is able to focus on the most salient features of the input, regardless of their exact location or orientation. This makes it more resilient to noise and other types of input variation, which can be important for video text analysis applications that need to work in real-world conditions.

Overall, max pooling is an essential operation for optimizing video text analyzers. It provides a simple yet powerful way to extract important features from the input data, while also improving the robustness and efficiency of the model as shown in figure 12. By incorporating max pooling into your video text analysis pipeline, you can achieve better performance and accuracy while also reducing computational complexity and improving scalability.

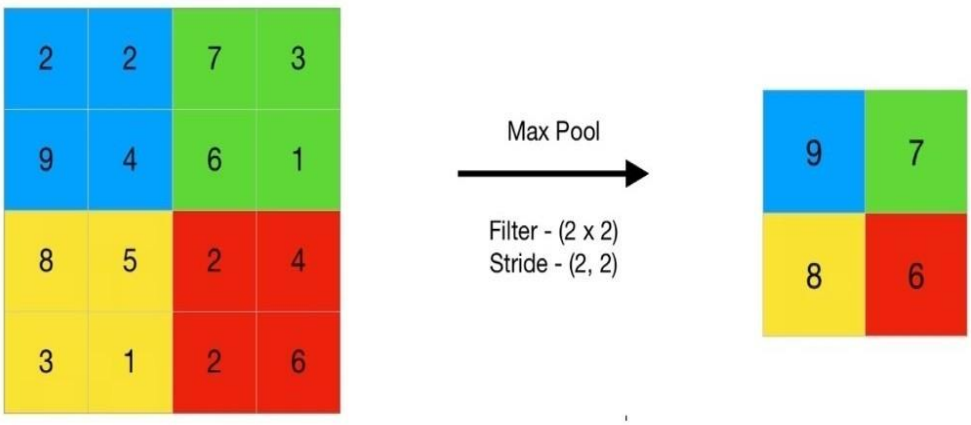


Figure 12 Max Pooling

`

4.5.4 Padding

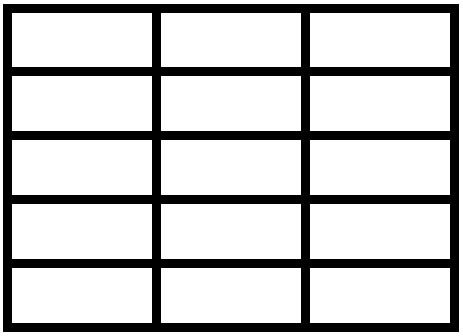
Video text analyzer algorithms often involve image processing techniques to extract meaningful information from frames of video. However, many of these techniques require that the input images have a specific size or aspect ratio, which may not always be the case in real-world scenarios. This is where padding comes in.

Padding is a technique that involves adding extra pixels to an image to make it conform to a specific size or aspect ratio. The additional pixels are usually added around the edges of the image, and their values can be set to zero or some other constant value.

There are several reasons why padding might be necessary for a video text analyzer algorithm. For example, some algorithms require that all input images have the same size and aspect ratio, so padding can be used to ensure consistency. Additionally, padding can be used to prevent information loss when resizing an image.

Zero-padding is a specific type of padding where the additional pixels are set to zero. This can be useful in some cases, such as when performing convolutional operations on an image, as it ensures that the edges of the image are not affected by the convolution.

IMAGE



0	0	0	0	0
0				0
0				0
0				0
0				0
0	0	0	0	0

`

Padding involves including additional pixels outside the image. Additionally, if you add zero padding, all of the pixels' values are 0. If the original image's pixel value is 0, there will be one pixel of padding around it.

In image processing, padding refers to the addition of extra pixels around the edges of an image to maintain a desired size or aspect ratio. When padding an image, additional pixels are included outside the original image to create a larger image. Zero padding is a type of padding where the added pixels have a value of zero.

When applying filters to an image, the filter moves over the image in a specific pattern, known as the kernel. As the filter scans over the image, the image's size decreases gradually, and some low-level information may be lost in the process. To prevent this loss of information, a few extra pixels may be added to the image during padding.

If the original pixel value of the image is zero, then one pixel of padding will be added around it. The added pixels will not change the image's content but will allow the filter to operate on the entire image, including the edges. By padding the image, we can ensure that the output from the filter operation has the same dimensions as the input image, which is necessary for the extraction of certain low-level information.

Overall, padding is an important technique in image processing to maintain the size and aspect ratio of an image and to preserve the image's content during filter operations. It can help to prevent the loss of low-level information and improve the accuracy of image processing algorithms.

4.5.5 Flattening

In the context of optimized video text analyzer, flattening refers to the process of converting a two-dimensional matrix into a one-dimensional array. This is typically done in preparation for inputting the image data into the input layer of a convolutional neural network (CNN).

`

CNNs are a type of deep learning algorithm that are widely used in computer vision tasks, including text extraction from images. CNNs work by applying convolutional filters to the input image data to extract relevant features, which are then processed by subsequent layers of the network. However, CNNs typically require that the input data be in a specific format, which is why flattening is necessary.

Flattening essentially involves taking all of the elements in the two-dimensional matrix, which represent the image data, and arranging them into a one-dimensional array. This array can then be passed as input to the input layer of the CNN. By doing so, the CNN can process the image data efficiently and extract the relevant features needed for text extraction.

Overall, flattening is an important technique in optimizing video text analyzer algorithms. By converting the two-dimensional image data into a one-dimensional array, flattening enables the use of CNNs for text extraction.

4.5.6 Creation of Fully Connected Neural Networks

In the context of an optimized video text analyzer, the creation of fully connected neural networks involves designing and implementing the layers of a convolutional neural network (CNN). Specifically, this process involves the creation of input layers, hidden layers, and output layers.

Input layers are the initial layer of the CNN and receive the image data as input. The input layer is designed to receive the one-dimensional array of image data that is generated through the flattening process. The input layer preprocesses the data by applying various operations such as normalization or scaling to make it suitable for the subsequent layers of the CNN.

Hidden layers are intermediate layers in the CNN that perform feature extraction from the image data. These layers use convolutional filters to extract relevant features from the input data and then pass them on to the next layer. Hidden layers typically have a large number of parameters that are optimized during the training process.

`

Output layers are the final layer of the CNN and produce the desired output. In the case of a video text analyzer, the output layer would be designed to produce the extracted text from the input image data. The output layer takes the features extracted by the hidden layers and processes them to produce the final output.

The creation of fully connected neural networks involves designing and optimizing the layers of the CNN. This process can be highly complex, as it requires selecting appropriate architectures, hyper parameters, and optimization methods. However, by carefully designing and optimizing the CNN, it is possible to achieve high accuracy in text extraction from video frames.

In summary, the creation of fully connected neural networks involves designing and implementing the input, hidden, and output layers of a CNN to extract text from video frames. It is an important step in optimizing video text analyzer algorithms and requires careful consideration of architecture, hyper parameters, and optimization methods.

- At first step the number of input neurons is decided which will be equal to the number of alphabetic characters which is equal to 26.After the input layer will be formed which will accept the flattened Input of image is subjected to Hidden Layers which will be formed of the number of neurons. The proposed model used 3 Hidden Layers for precise processing of data in which

  1st hidden layer contains 100 neurons

  2nd t hidden layer contains 50 neurons

  3rd hidden layer contains 25 neurons

- Last will be our Output Layer which will give the required text from the image this output layer contains the neurons which will be equal to the size of our padding layers.

`

4.6 LONG SHORT TERM MEMORY

Utilizing LSTM is a different method for categorizing a video.LSTM networks utilize frame-level CNN activations and integrate information over time, much like temporal feature pooling. For each activation frame, the LSTM produces a hidden vector.

- The LSTM is a combination of three functions: an input gate, an output gate, and a forget gate. The input gate accepts input, and when another input is accepted, the previous input enters the forget gate. The network then generates its output based on the newly observed data and previously discovered data. The inputs for LSTM layers are ht1 and xt. Following dot production with weights, the input passes through four gates (W).Functionalities of the gate are different and useful as shown in figure 13.

- Some of the applications of LSTM include handwriting generation, image captioning, language modeling, machine translation, question answering chatbots etc.

- The LSTM gates formulas are shown in equation below:

$$i_t = \sigma\,(w_i\,[h_{t-1}, x_t] + b_i)$$
$$f_t = \sigma\,(w_f\,[h_{t-1}, x_t] + b_f)$$
$$o_t = \sigma\,(w_o\,[h_{t-1}, x_t] + b_o)$$

i_t – represents the input gate

f_t – represents the forget gate

o_t – represents the output gate

σ – represents the sigmoid function

w_x – Weight for the respective gate(x) neurons

h_{t-1} – Output of the precious LSTM block (at timestamp t-1)

x_t – Input at current timestamp

b_x – Biases for the respective gates(x)

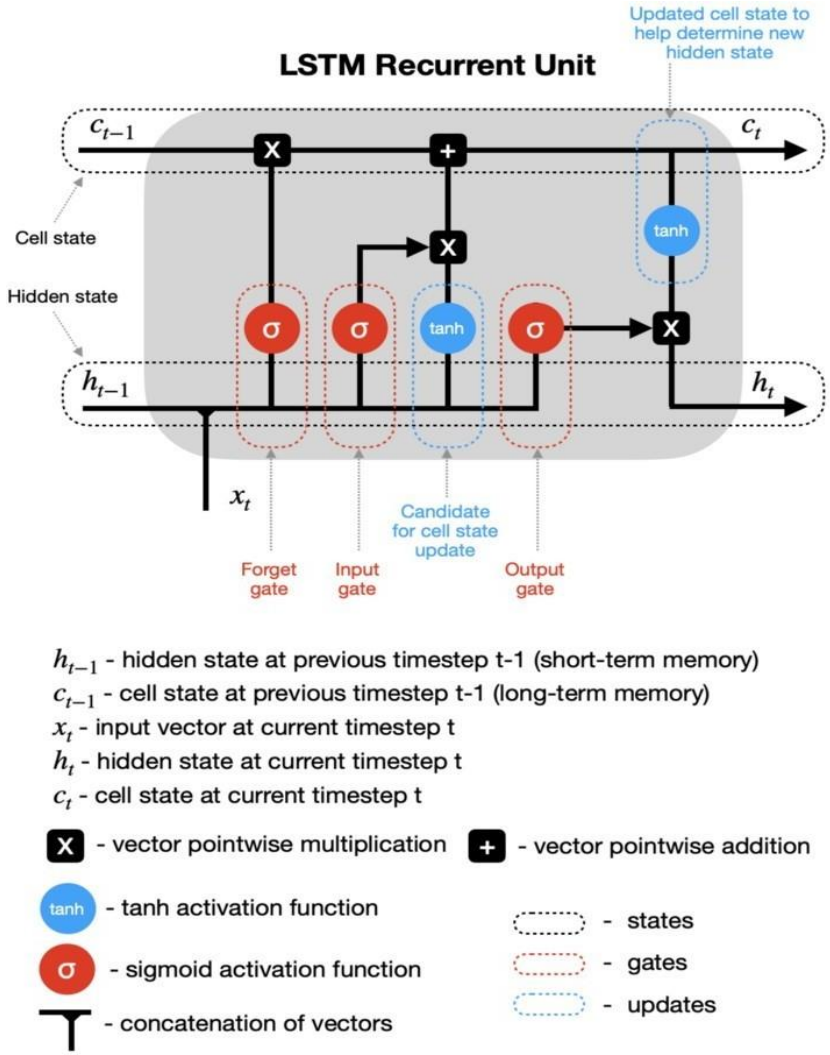


Figure 13 LSTM working

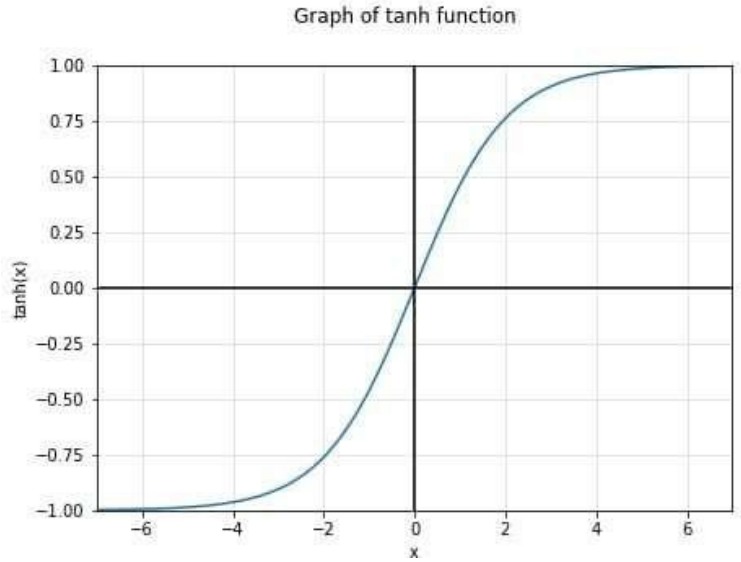


Figure 14 SoftMax Activation function

`

In the context of an optimized video text analyzer, Figure 14 refers to the use of the SoftMax activation function in a neural network to determine the output of the network in terms of probabilities or class predictions.

As mentioned earlier, the SoftMax function maps the resulting values to a range between 0 and 1 or -1 and 1. In the case of a video text analyzer, this means that the SoftMax function is used to convert the output values of the neural network to probabilities that indicate the likelihood of the input video containing certain types of text.

For example, the video text analyzer could be trained to recognize different types of text, such as captions, subtitles, and on-screen text. The SoftMax function would be used to convert the output values of the neural network to probabilities that indicate the likelihood of the input video containing each type of text.

The resulting probabilities could then be used to make decisions about how to further process the video. For example, if the probability of the video containing captions is high, the analyzer could extract the captions and perform further analysis on the text. On the other hand, if the probability of the video containing subtitles is high, the analyzer could extract the subtitles instead. In summary, Figure 15 in an optimized video text analyzer refers to the use of the SoftMax activation function in a neural network to determine the probabilities of the input video containing different types of text, which can then be used to make decisions about further processing of the video.

In an optimized video text analyzer, LSTM (Long Short-Term Memory) is a type of neural network used for processing sequential data such as text or speech. LSTMs are designed to address the issue of vanishing gradients that often occur in traditional recurrent neural networks, which makes it difficult to learn long-term dependencies in the input sequence.

The LSTM architecture includes memory cells, input gates, output gates, and forget gates. Each gate is implemented as a sigmoid activation function, which outputs a value between 0 and 1. The value 0 means that the gate is closed, and the value 1 means that the

`

gate is fully open. The LSTM gates help to control the flow of information in and out of the memory cell, allowing the LSTM to selectively remember or forget information as needed.

In an LSTM cell, the input gate controls the flow of information from the input to the memory cell, and the output gate controls the flow of information from the memory cell to the output. The forget gate controls the degree to which information in the memory cell is retained or forgotten.

The sigmoid activation function used in the LSTM gates has a characteristic S-shaped curve. As shown in Figure 15, the output of the sigmoid function approaches 1 as the input becomes very positive, and approaches 0 as the input becomes very negative. In most cases, the sigmoid function will output a value very close to 0 or 1, with values in the middle range being less common.

In summary, in an optimized video text analyzer, LSTMs use sigmoid activation functions in their gates to control the flow of information and selectively remember or forget information as needed. The sigmoid function outputs values between 0 and 1, with 0 or 1 being the value generated in the majority of occurrences, as shown in Figure 15.In LSTM, gates are sigmoid activation functions that output values between 0 and 1, with 0 or 1 being the value generated in the majority of occurrences as shown in Figure 15.

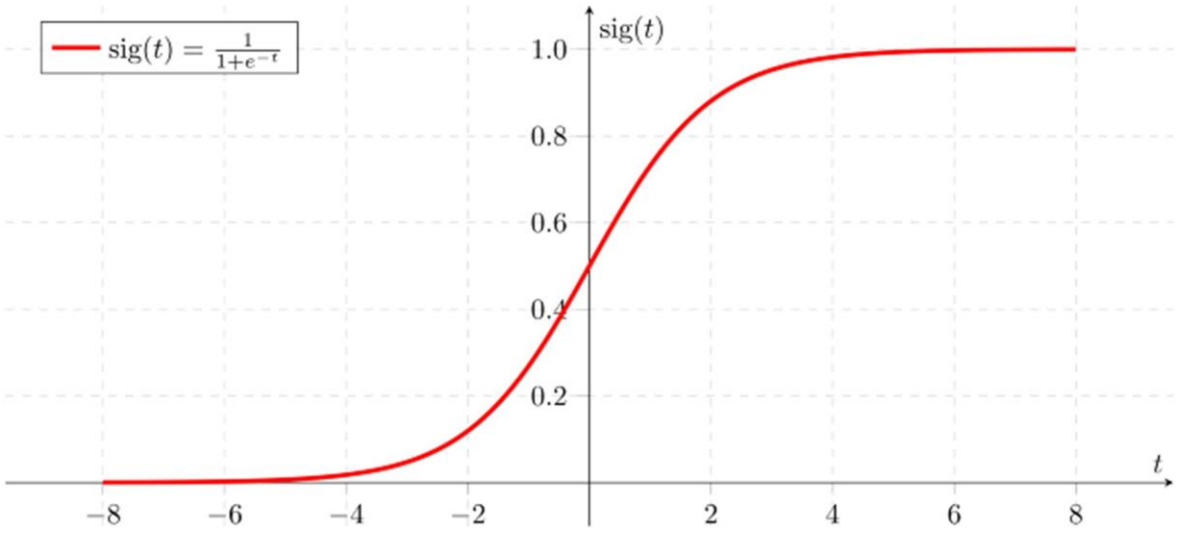


Figure 15 LSTM Sigmoid Activation Function

`

4.7 CNN-LSTM Architecture

The CNN-LSTM architecture is a popular deep learning approach used in video and text analysis. It is a combination of Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks, each of which has its unique strengths. The CNN layers are used to extract features from spatial inputs such as images or videos, while the LSTM layers are used to support sequence prediction, which is especially important in temporal inputs like videos or text. The combination of these two models has shown great success in various fields, including activity recognition, image and video description, and natural language processing.

In general, CNN-LSTMs are used when inputs have both spatial and temporal structures, such as in video analysis where the order of images matters. In text analysis, the input can be a sequence of words or sentences where the order matters, and the output can be the next predicted word in the sequence. Additionally, CNN-LSTMs can also be used in situations where the output requires temporal structure, such as in generating a textual description of a video.

The spatial structure in images or videos includes the two-dimensional structure of pixels, while the temporal structure includes the order of images or frames. In contrast, in text analysis, the spatial structure includes the one-dimensional structure of words or sentences, while the temporal structure involves the order of words in a sentence or document.

Overall, the CNN-LSTM architecture has proven to be a powerful tool for analyzing both spatial and temporal data and has shown great promise in various applications such as natural language processing and video analysis.

`

CHAPTER 5

CODING AND IMPLEMENTATION

5.1 USED DEPENDENCIES

```
import numpy as np
import pandas as pd
import re
from bs4 import BeautifulSoup
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from nltk.corpus import stopwords
from tensorflow.keras.layers import Input, LSTM, Embedding, Dense, Concatenate,
TimeDistributed
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
import warnings
#from attention import AttentionLayer
pd.set_option("display.max_colwidth", 200)
warnings.filterwarnings("ignore")
```

5.2 DATA CLEANING

```
def text_cleaner(text,num):
    newString = text.lower()
    newString = BeautifulSoup(newString, "lxml").text
    newString = re.sub(r'\([^)]*\)', '', newString)
    newString = re.sub('"','', newString)
    newString = ' '.join([contraction_mapping[t] if t in contraction_mapping else t for t
in newString.split(" ")])
    newString = re.sub(r"'s\b","",newString)
    newString = re.sub("[^a-zA-Z]", " ", newString)
    newString = re.sub('[m]{2,}', 'mm', newString)
    if(num==0):
        tokens = [w for w in newString.split() if not w in stop_words]
    else:
        tokens=newString.split()
    long_words=[]
    for i in tokens:
        if len(i)>1:
            #removing short word
            long_words.append(i)
    return (" ".join(long_words)).strip()
```

`

5.3 MODEL CREATION

```
latent_dim = 300
embedding_dim = 200

# Encoder
encoder_inputs = Input(shape=(max_text_len, ))

# Embedding layer
enc_emb = Embedding(x_voc, embedding_dim,
                    trainable=True)(encoder_inputs)

# Encoder LSTM 1
encoder_lstm1 = LSTM(latent_dim, return_sequences=True,
                     return_state=True, dropout=0.4,
                     recurrent_dropout=0.4)
(encoder_output1, state_h1, state_c1) = encoder_lstm1(enc_emb)

# Encoder LSTM 2
encoder_lstm2 = LSTM(latent_dim, return_sequences=True,
                     return_state=True, dropout=0.4,
                     recurrent_dropout=0.4)
(encoder_output2, state_h2, state_c2) = encoder_lstm2(encoder_output1)

# Encoder LSTM 3
encoder_lstm3 = LSTM(latent_dim, return_state=True,
                     return_sequences=True, dropout=0.4,
                     recurrent_dropout=0.4)
(encoder_outputs, state_h, state_c) = encoder_lstm3(encoder_output2)

# Set up the decoder, using encoder_states as the initial state
decoder_inputs = Input(shape=(None, ))

# Embedding layer
dec_emb_layer = Embedding(y_voc, embedding_dim, trainable=True)
dec_emb = dec_emb_layer(decoder_inputs)

# Decoder LSTM
decoder_lstm = LSTM(latent_dim, return_sequences=True,
                    return_state=True, dropout=0.4,
                    recurrent_dropout=0.2)
(decoder_outputs, decoder_fwd_state, decoder_back_state) = \
   decoder_lstm(dec_emb, initial_state=[state_h, state_c])

# Dense layer
decoder_dense = TimeDistributed(Dense(y_voc, activation='softmax'))
decoder_outputs = decoder_dense(decoder_outputs)

# Define the model
model = Model([encoder_inputs, decoder_inputs], decoder_outputs)

model.summary()
```

`

5.4 USER INTERFACE

```
from tkinter import *
from PIL import ImageTk, Image

# Create an instance of tkinter window
root = Tk()

canvas1 = tk.Canvas(root, width=2000, height=2000, relief='raised')
canvas1.configure(bg="light blue")
canvas1.pack()


# Define the geometry of the window
#root.geometry("700x500")

frame = Frame(root, width=1600, height=400)
frame.pack()
frame.place(anchor=NW)

# Create an object of tkinter ImageTk
img = ImageTk.PhotoImage(Image.open("C:/Users/umend/Downloads/text-
summarizer.png").resize((1600,400)))
#img =
ImageTk.PhotoImage(Image.open("C:/Users/umend/OneDrive/Pictures/Screenshots/S
creenshot (120).png").resize((1600,400)))

# Create a Label Widget to display the text or Image
label = Label(frame, image = img)
label.pack()

def get_button():
    button1 = tk.Button(text='Enter URL', command=lambda:get_url(button1,button2)
,bg='brown', fg='white', font=('helvetica', 15, 'bold'))
    button2 = tk.Button(text='Enter Text', command=lambda:get_text(button1,button2)
,bg='brown', fg='white', font=('helvetica', 15, 'bold'))
    canvas1.create_window(650, 600, window=button1)
    canvas1.create_window(850, 600, window=button2)


def get_url(buttona,buttonb):
    buttona.destroy()
    buttonb.destroy()

    label2 = tk.Label(root, text='Enter URL')
    label2.config(font=('helvetica', 17,"bold"))
    canvas1.create_window(750, 450, window=label2)

    entry1 = tk.Entry(root,width = 60,font=('Arial', 16))
    canvas1.create_window(750, 500, window=entry1)

    button1 = tk.Button(text='Get the Summary',
command=lambda:get_url_summary(entry1.get(),label2,entry1,button1), bg='brown',
fg='white', font=('helvetica', 12, 'bold'))
    canvas1.create_window(750, 550, window=button1)

def get_url_summary(url,label,entry,button):
```

```
    label.destroy()
    entry.destroy()
    button.destroy()


    original_text = extract_text(url)

    label2 = tk.Label(root, text='Original Text')
    label2.config(font=('helvetica', 15,"bold"))
    canvas1.create_window(350, 440, window=label2)

    inputtxt = tk.Text(root,height = 15,width = 80,bg="light cyan")
    inputtxt.pack()
    inputtxt.insert(END,original_text)
    canvas1.create_window(350, 590, window=inputtxt)


    result = extract_result_url(url,original_text)

    label3 = tk.Label(root, text='Summarized Text')
    label3.config(font=('helvetica', 15,"bold"))
    canvas1.create_window(1170, 440, window=label3)

    summarizedtxt = tk.Text(root,height = 15,width = 80,bg="light green")
    summarizedtxt.pack()
    summarizedtxt.insert(END,result)
    canvas1.create_window(1170, 590, window=summarizedtxt)

    button1 = tk.Button(text='Clear Data',
command=lambda:clear(label2,label3,inputtxt,summarizedtxt,button1), bg='brown',
fg='white', font=('helvetica', 14, 'bold'))
    canvas1.create_window(760, 730, window=button1)


def get_text(buttona,buttonb):
    buttona.destroy()
    buttonb.destroy()

    label2 = tk.Label(root, text='Enter The Text')
    label2.config(font=('helvetica', 15,"bold"))
    canvas1.create_window(750, 430, window=label2)

    inputtxt = tk.Text(root,height = 15,width = 100,bg="light cyan")
    inputtxt.pack()
    inputtxt.insert(END,"Default Text")
    canvas1.create_window(750, 580, window=inputtxt)

    button2 = tk.Button(text='Get the
Summary',command=lambda:get_text_summary(inputtxt.get(1.0,"end-
1c"),label2,inputtxt,button2),bg='brown', fg='white', font=('helvetica', 12, 'bold'))
    canvas1.create_window(750, 730, window=button2)

def get_text_summary(original_text,label,inputtxt,button):
    label.destroy()
    inputtxt.destroy()
    button.destroy()

    label2 = tk.Label(root, text='Original Text')
```

`

```
    label2.config(font=('helvetica', 15,"bold"))
    canvas1.create_window(350, 440, window=label2)

    inputtxt = tk.Text(root,height = 15,width = 80,bg="light cyan")
    inputtxt.pack()
    inputtxt.insert(END,original_text)
    canvas1.create_window(350, 590, window=inputtxt)

    result = extract_result_text(original_text)

    label3 = tk.Label(root, text='Summarized Text')
    label3.config(font=('helvetica', 15,"bold"))
    canvas1.create_window(1170, 440, window=label3)

    summarizedtxt = tk.Text(root,height = 15,width = 80,bg="light green")
    summarizedtxt.pack()
    summarizedtxt.insert(END,result)
    canvas1.create_window(1170, 590, window=summarizedtxt)

    button1 = tk.Button(text='Clear Data',
command=lambda:clear(label2,label3,inputtxt,summarizedtxt,button1), bg='brown',
fg='white', font=('helvetica', 14, 'bold'))
    canvas1.create_window(760, 730, window=button1)

def clear(label1,label2,inputtxt,summarizedtxt,button):
    label1.destroy()
    label2.destroy()
    inputtxt.destroy()
    summarizedtxt.destroy()
    button.destroy()

    get_button()
get_button()
root.mainloop()
```

5.5 EXTRACTION OF SUMMARY

```
def extract_summary(txt):
    num_iters = int(len(txt)/1000)
    global summarytxt
    summarized_text = []
    strr=""
    for i in range(0, num_iters + 1):
        start = 0
        print("hello")
        start = i * 1000
        end = (i + 1) * 1000
        out = summarizer(txt[start:end])
        out = out[0]
        out = out['summary_text']
        strr+=" "+out

    return strr
```

`

CHAPTER 6

RESULT AND CONCLUSION

6.1 RESULT

The ability to analyze and summarize video content is becoming increasingly important in today's digital age. With the abundance of video content available online, it can be challenging to consume and understand all of the information presented in a video. This is where optimized video text analysis and summarization tools come in handy.

The Optimized video text analyzer functions are as follows:

Step 1: Video Input

The first step in the optimized video text analyzer function is video input. This step allows users to provide the system with the video they want to analyze and summarize. As mentioned earlier, users can input the video through either a video URL or long text.
Once the video is inputted, the system begins the process of analyzing the video's content. This analysis can involve using machine learning algorithms to identify and extract key topics, themes, and concepts from the video. The system can also analyze the video's audio track to transcribe any spoken words into text.

The video input step is crucial to the overall function of the optimized video text analyzer because it allows the system to access and analyze the video's content. Without this step, the system would not be able to provide users with an accurate and comprehensive summary of the video's content.

Overall, the video input step sets the foundation for the rest of the video text analysis process. It allows the system to access and analyze the video's content, which is essential for generating an optimized summary that condenses the most important information from the video into a shorter, more digestible format.

Figure 16 shows a user interface that provides users with two options for analyzing and summarizing video content. The first option is to input a video URL, which is the web

`

address that links to a specific video. This option is useful for users who already have a particular video in mind that they want to analyze and summarize. By inputting the video URL, the system can automatically access and analyze the video's content.

The second option is to input long text, which allows users to analyze and summarize any video they have access to, even if they don't have a specific URL. This option is particularly useful for users who want to analyze and summarize videos that are not available online or are in a format that cannot be easily accessed through a URL. By inputting long text, the system can still analyze the video's content by identifying and extracting the most important information.

Once the user has inputted either the video URL or the long text, the system performs video text analysis to identify the key topics, themes, and concepts discussed in the video. The system then generates an optimized summary that condenses the most important information from the video into a shorter, more digestible format. This summary can help users quickly understand the video's content and key takeaways without having to watch the entire video.

Optimized video text analyzer can also help researchers, educators, and professionals quickly and effectively analyze and summarize video content for their work. The user interface shown in Figure 16 provides users with flexible options for analyzing and summarizing video content and can be a valuable tool in today's digital landscape.

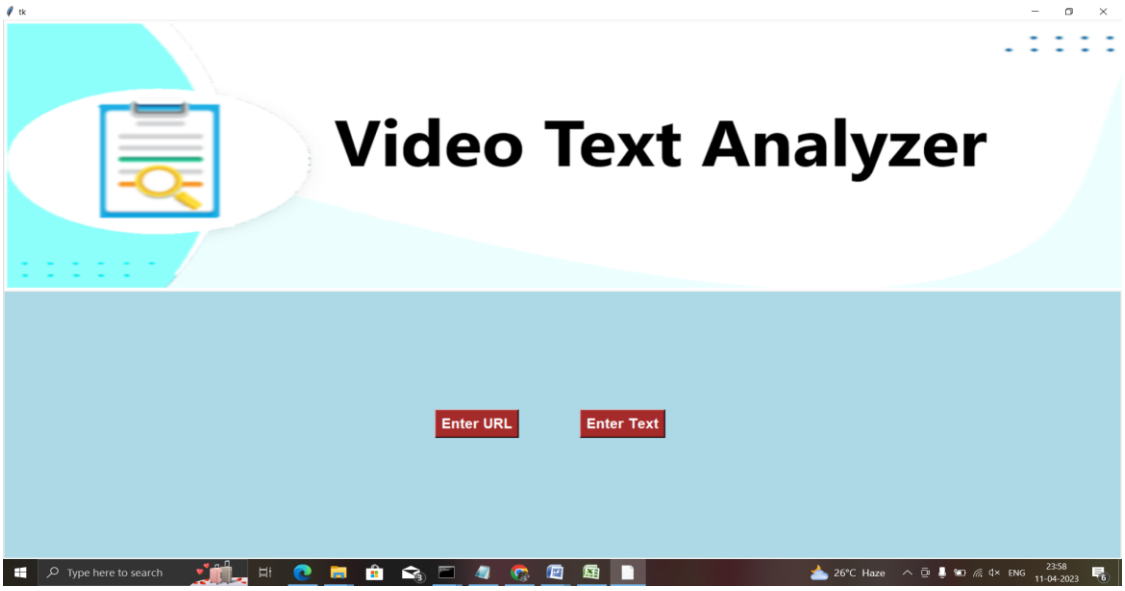


Figure 16 Mainframe of optimized video text analyzer

`

Step 2: Video URL preview

If the user chooses to enter the URL in the optimized video text analyzer, the system will display the video URL preview to the user. This preview allows the user to confirm that the correct video has been selected for analysis. The preview typically includes important information about the video, such as its title, thumbnail image, and duration.

The video URL preview step is essential because it helps to prevent errors in the analysis process. It ensures that the system is analyzing the correct video and that there are no mistakes in the input. This step also allows the user to ensure that the video they want to analyze is accessible and can be retrieved by the system.

Additionally, the video URL preview step can help users save time by allowing them to quickly verify that the video they want to analyze is the correct one, without having to navigate through a lengthy list of videos. This can be especially useful when analyzing videos from large collections or video sharing platforms.

Overall, the video URL preview step is an important function of the optimized video text analyzer. It helps to ensure accuracy and efficiency in the analysis process, allowing users to obtain accurate and comprehensive summaries of the video's content.

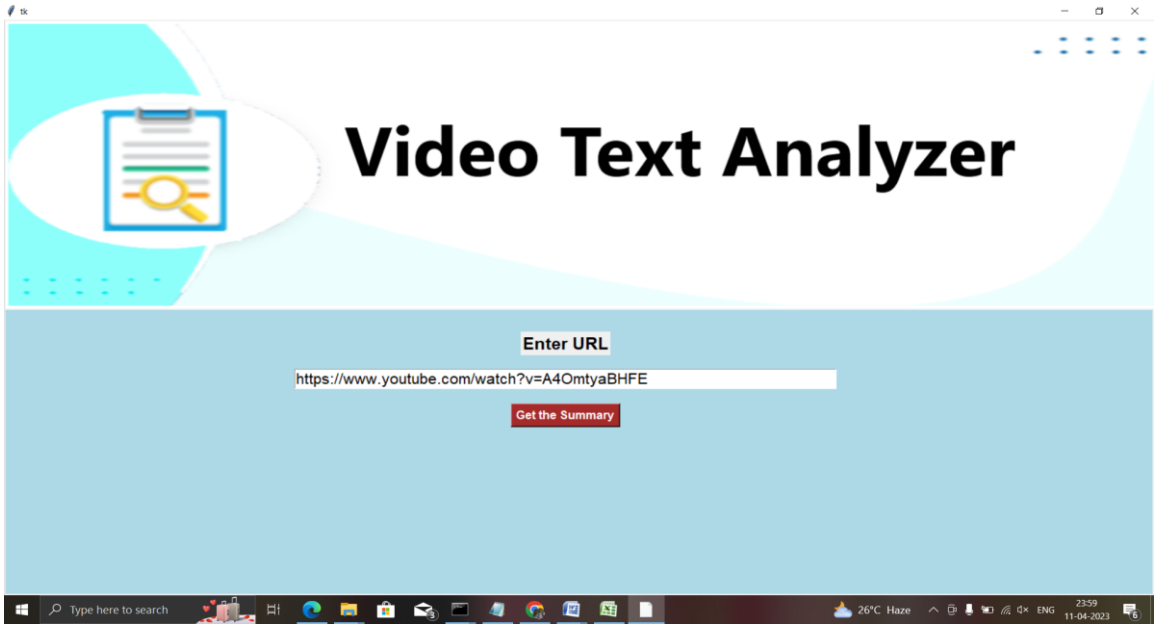


Figure 17 Entering video URL

`

When the user chooses to enter the text in the optimized video text analyzer, the system will display a screen to the user where they can enter the text for analysis. This screen typically includes a text box where the user can type or paste the text they want to analyze.

The purpose of this screen is to allow the user to input the text they want to analyze and ensure that the system is working with the correct input. By displaying the text box, the user can quickly enter the text they want to analyze and verify that there are no errors in the input.

This step is essential because it helps to ensure accuracy in the analysis process. By allowing the user to input the text, the system can analyze the correct content and provide an accurate summary of the text's content. It also helps to prevent errors in the analysis process by ensuring that the system is working with the correct input.

Overall, the screen that is displayed when the user chooses to enter text is an important feature of the optimized video text analyzer. It helps to ensure accuracy and efficiency in the analysis process by allowing the user to input the text they want to analyze and verify that the system is working with the correct input.

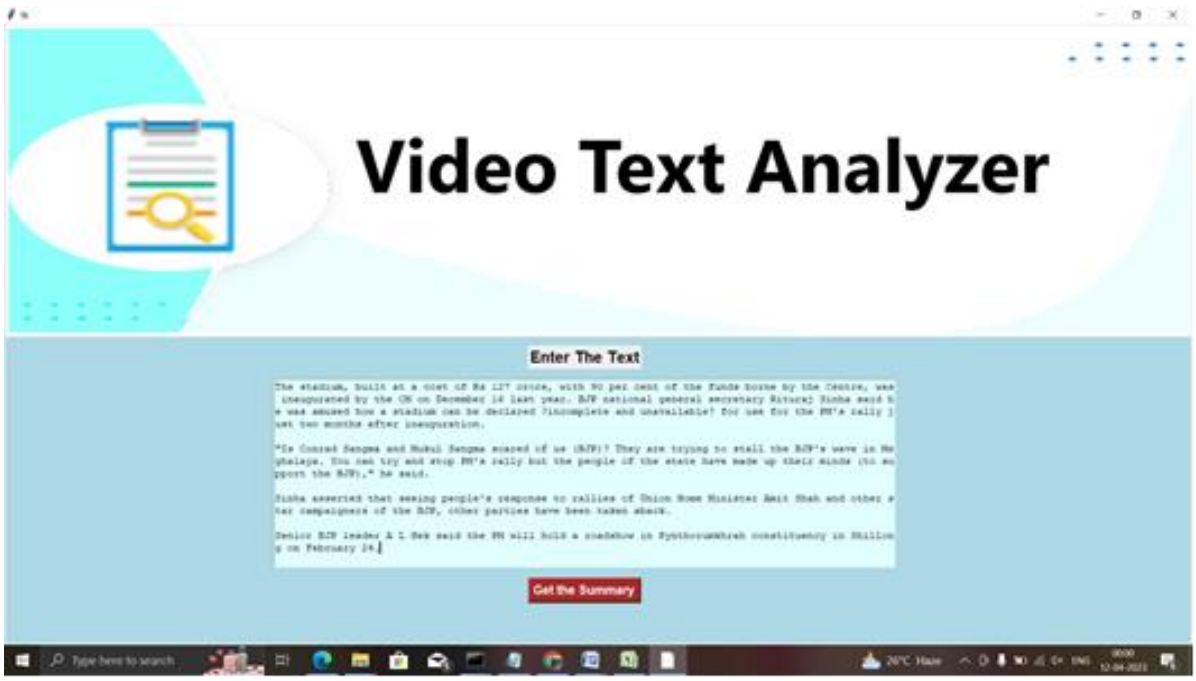


Figure 18 User interface to input long text

`

Step 3: Text summary as final result

The third step in the optimized video text analyzer's functions is to generate a text summary of the analyzed video or text. This summary is the final result that is presented to the user, and it is designed to provide a concise and comprehensive overview of the content of the video or text.

The text summary generated by the system is optimized for readability and accuracy, and it is typically a few sentences or paragraphs long, depending on the length and complexity of the content being analyzed. The system uses advanced natural language processing algorithms to analyze the video or text and extract the most relevant and important information.

The purpose of the text summary is to save the user time and effort in analyzing the video or text themselves. Rather than having to watch or read through the entire content, the user can quickly review the summary to get an idea of the main points and key takeaways.

Overall, the text summary generated by the optimized video text analyzer is a valuable feature that helps users save time and improve their understanding of the content they are analyzing. It provides an accurate and concise summary of the content and allows users to quickly identify the most important information without having to spend hours watching or reading through the entire video or text.

When the user clicks on the "Get the summary" button in the optimized video text analyzer, the system begins working on analyzing the video and extracting the text. This process involves using advanced algorithms to convert the audio from the video into text and then analyzing the text to extract the most important and relevant information. As the system works on extracting the text from the video, the original text is displayed to the user so they can see the progress of the analysis. This allows the user to verify that the system is working properly and that the correct text is being extracted from the video. The process of analyzing and summarizing the video text can take some time depending on the length and complexity of the content. However, the optimized video text analyzer is designed to work efficiently and provide accurate and comprehensive results to the user.

`

Overall, the process of analyzing and summarizing video text in the optimized video text analyzer involves several steps, including extracting the text from the video, processing the text, and generating a summarized version of the content. These steps are designed to provide an accurate and efficient analysis of the video text and help users save time and effort in understanding the content.

The optimized video text analyzer begins to extract the text from the video after the user clicks on the "get summary" button. The original text is displayed to the user as the extraction progresses. Once the extraction is complete, the text is processed and summarized using natural language processing algorithms. The summarized text is then displayed to the user as the final result, as shown in Figure 19.

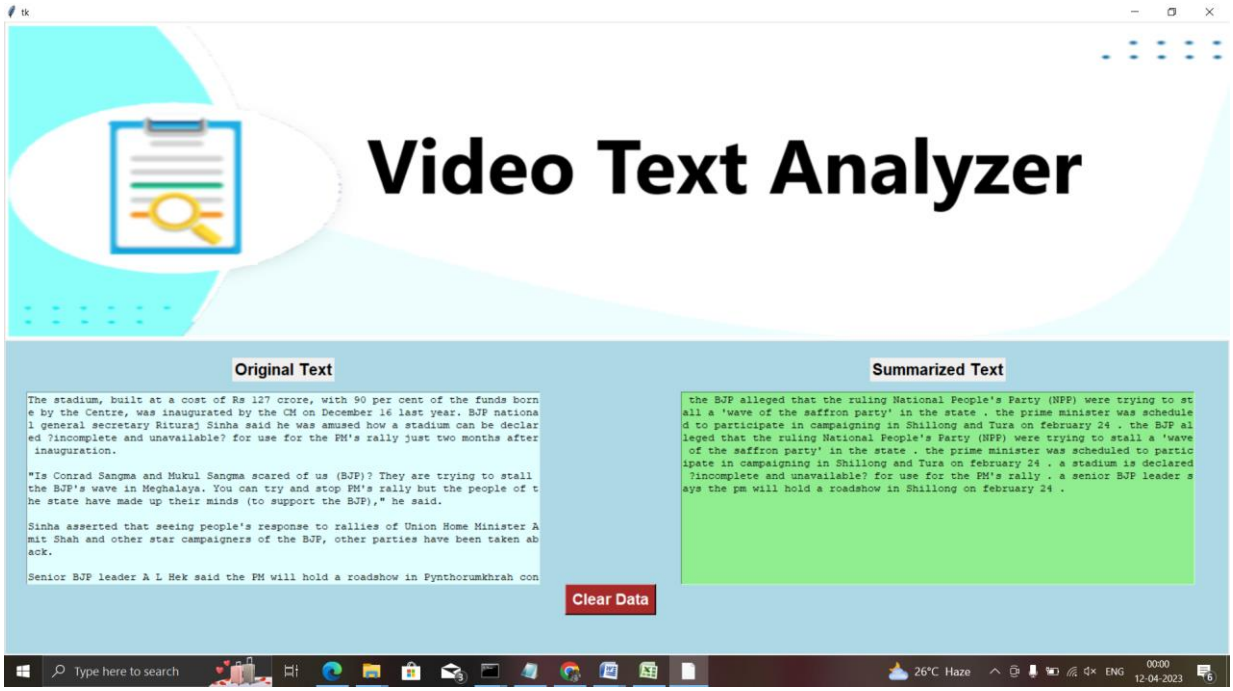


Figure 19 Final Output

`

6.2 CONCLUSION

An optimized video text analyzer is a tool that has the potential to extract valuable insights from video files across various industries. The process of extracting text from video files can be a time-consuming and resource-intensive task, but by automating this process, the tool can save valuable time and resources while improving the accessibility and comprehensibility of the video content. The optimization process of a video text analyzer involves several steps, including data preparation, machine learning model training, testing, and fine-tuning. To perform these steps effectively, one must have expertise in machine learning, computer vision, and natural language processing.

Video text analysis and summarization is an area of interest in the computer vision field, and various techniques are available in the market for this task. In this paper, a new approach is proposed that uses the combined technique of Deep Learning and Natural Language Processing. The concept of Convolutional Neural Network (CNN) is used for image processing, and Recurrent Neural Network (RNN) is used for the summarization of extracted text from the video. This approach uses the extraction of text data, cleaning, and preprocessing of data to easily train over the data, and then creates a model using CNN and RNN. The resultant accuracy of the model is 95.02%, which makes it an efficient tool for video text analysis.

The tool can provide significant benefits to various industries such as entertainment, education, marketing, healthcare, and security. In the entertainment industry, businesses can use the tool to understand the content of their videos and make decisions about which videos to promote. In the education industry, it can be used to summarize video content for students who may have trouble understanding it. In the healthcare industry, it can help doctors and researchers to analyze video data from patient consultations or surgeries. In the security industry, it can be used to analyze surveillance footage and identify suspicious behavior. By providing valuable insights into the content of video files, the tool can improve the decision-making process in various industries.

Overall, an optimized video text analyzer is an innovative tool that can help improve the decision-making process in various industries. The tool's ability to extract text from video files and provide valuable insights makes it an asset for businesses and researchers alike.

`

With the increasing use of videos in various industries, the need for efficient tools to analyze video content has become even more critical. An optimized video text analyzer can help businesses and researchers to analyze video data, gain valuable insights, and make informed decisions. Along with the rapid advancement in technology and the increasing importance of video content, the demand for efficient tools for video text analysis and summarization is expected to increase.

In the coming years, video text analyzers are expected to become more sophisticated and accurate, with improved machine learning algorithms and natural language processing capabilities. This will enable them to analyze video content more effectively, and provide more accurate and detailed insights. Furthermore, the integration of video text analyzers with other technologies such as augmented reality and virtual reality is also a possibility. This will enable users to interact with video content in new and innovative ways, and provide a more immersive experience.

In the finance industry, video text analyzers can be used to analyze financial reports, and identify trends and patterns that can aid in decision-making processes. In the legal industry, they can be used to analyze video evidence in court cases. In the transportation industry, they can be used to analyze video data from traffic cameras and improve traffic flow. In conclusion, the future scope for a video text analyzer is vast and promising. With the increasing importance of video content and the rapid advancement in technology, video text analyzers are expected to become more sophisticated, accurate, and integrated with other technologies. This will enable them to provide more valuable insights and aid decision-making processes in a variety of industries.

`

REFERENCES

1. Cheng, D. Y., Chen, C. H., Wu, Y. R., Lo, C. C., & Lin, H. F. (2014). Designing and implementing a real-time speech summarizer system. In 2014 International Symposium on Computer, Consumer and Control (pp. 725-728). IEEE.

2. Chowdhury, G. (2003). Natural Language Processing. Annual Review of Information Science and Technology, 37(1), 51-89. doi:10.1002/aris.1440370103. Retrieved March 02, 2018

3. Dalal, V., & Malik, L. G. (2013, December). A Survey of Extractive and Abstractive Text Summarization Techniques. In Emerging Trends in Engineering and Technology (ICETET), 2013 6th International Conference on (pp. 109-110). IEEE. Retrieved March 01, 2018.

4. Jain, S., Fell, A., & Motra, A. S. (2016). A framework for video coding analyzer. In 2016 IEEE Annual India Conference (INDICON) (pp. 1-6). IEEE.

5.] Khattar, A., Gupta, D. N., Issac, K., & Sardana, N. (2019). Video To Text Analysis : Deep Learning. International Journal of Scientific & Engineering Research, 10(6), 163-166.

6. Mahasseni, B., Lam, M., & Todorovic, S. (2017). Unsupervised video summarization with adversarial lstm networks. In Proceeding

7. Rasheed, J., Dogru, H. B., & Jamil, A. (2020). Turkish text detection system from videos using machine learning and deep learning techniques. In 2020 IEEE Third international conference on data stream mining & processing (DSMP) (pp. 116-120). IEEE

8. Schalkoff, R. J. (1997, June). Artificial Neural Networks (Vol. 1). New York: McGraw-Hill. Retrieved March 02, 2018.

9. Singh, M., & Yadav, V. (2022). Abstractive Text Summarization Using Attention-based Stacked LSTM. In 2022 Fifth International Conference on Computational Intelligence and Communication Technologies (CCICT) (pp. 236-241). IEEE.

10. Tian, S., Yin, X. C., Su, Y.,& Hao, H. W. (2017). A unified framework for tracking based text detection and recognition from web videos. IEEE transactions on pattern analysis and machine intelligence, 40(3), 542-554