



# Video Subtitle Generator for Enhanced Digital learning

**Zeon Gouria (24)**

**Kevin Jadhav (25)**

**Sam Jebaraj (39)**

**Yash Malhi (74)**

**Supervisor:**

**Prof. Nilambari Narka**  
**Computer Engineering Department**  
Xavier Institute of Engineering  
University Of Mumbai

## **Abstract**

Online video lecture repositories are rapidly growing because of this pandemic and becoming established as fundamental knowledge assets. However, most lectures are neither transcribed nor translated due to the shortage of cost-effective solutions which will give accurate enough results. The primary objective of developing this system is to create an automated way to generate the subtitles for audios and videos. By replacing the tedious method of the current system of listening and typing the subtitles out, our system will save time, reduce the amount of work the administration has to do and will generate the subtitles automatically with proper application tools.

In this project we developed an integrated framework of automatic bilingual subtitle generation for lecture videos, especially for MOOCs (Massive Open Online Courses). Video plays an important role to assist people understand and comprehend the knowledge for instance the songs, movies or the video lectures or the other multimedia data relevant to the user. Hence to remove the gaps of their native language. This can be best done by the utilization of subtitles of the video.

It will contain meaningful material so as to supply an appropriate experimental program. Subtitle generator is going to deepen our knowledge of media video and audio, voice recognition and time synchronization. Besides, it'll provide resources for the aim of selecting the foremost convenient programming language to be used to develop the experimental program also as samples to know Application programming interface (API) features.

# Chapter 1

## Introduction

Video plays a vital role to help people understand and comprehend the information for example the songs, movies or the video lectures or any other multimedia data relevant to the user. Hence, here it becomes important to make videos available to the people having auditory problems and even more for the people to remove the gaps of their native language. This can be best done by the use of subtitles of the video. However, downloading subtitles of any video from the internet is a monotonous process. Consequently, to generate subtitles automatically through the software itself and without the use of internet is a valid subject of research. Online video lecture repositories are rapidly growing and becoming established as fundamental knowledge assets. However, most lectures are neither transcribed nor translated because of the lack of cost-effective solutions that can give accurate enough results.

### 1.1 Problem Definition

Since the pandemic everything has become digital, so as the learning process for all lectures and courses have become digital, so people with hearing impairment or Auditory processing disorder (APD), also known as central auditory processing disorder (CAPD), is a complex problem affecting about 5 percent of school-aged children. These kids can't process the information they hear in the same way as others because their ears and brain don't fully coordinate. So as things have gone online these students start to face issues in learning. Digital learning also comes up with other issues like low audio, corrupted audio file or damaged sound output devices. These problems can be solved with a subtitle. Hence we propose a system to generate a subtitle for a video which will help students with various issues with audio video files.

### 1.2 Aims and Objectives

#### 1.2.1 Aim:

To generate subtitle for video, for digital learning enhancement.

#### 1.2.2 Objectives:

- Perform an initial survey of the existing work to know its limitations.
- To develop a project using latest and fastest technology.
- Work on additional features to make it more appealing and user-friendly

### 1.3 Scope of the Project

The project will generate subtitle for videos using the developed models. We are developing the project to speed up the time required to generate a subtitle and also synchronize the following output.

## 1.4 Existing System

There are many system designed for subtitle generator which uses different system. One of the major system used is Uses Speech Recognition tool by CMU, DeepSpeech: This is a Speech Recognition tool by Mozilla, which uses Long Short-Term Memory (LSTM) for speech recognition. The other system that exist is FFMPEG libraries, Speech recognition modules. These two are one of the major system used for generating subtitle.

# Chapter 2

## Review of Literature

### 2.1 Subtitle Generation using CMU SPINHX

In the subsequent sections, we will see how to generate music using various algorithms. Now, let's take a look at how SPINHX are used to generate subtitle. Specifically, we are elaborating on the paper here we are studying on the approach to generate subtitle using Sphinx module for speech recognition. The audio is first extracted and then passed through speech recognition engines which transcribes the audio and then given us the desired. The specific engines used to transcribes the audio is cmu shinx. CMU sphinx uses Hidden Markov Model (HMM). The data set that is provided to train the following engines is Mozilla Common Voice data set this data set is chosen as it has a set of different speaker and more than 1000 hours of recorded audio of different speakers having different accents. Sphinx uses an HMM based model to help recognize a given audio input. We know a HMM works based on transition probabilities between entities to arrive at an overall probability of reaching a given state. In the case of Sphinx, the so-called inputs against which the transition probability matrix is evaluated is called "phones". These Phones are parts of speech that come from the pronunciation of different words. Studies show that the English language has 40 distinct phones which means words in the English language can be uttered by a combination of these 40 phones. It is a really big computational task when we would have to find the best match for a sequence of phones that represent a word. There are 3 independent models that are used in tandem to solve the speech recognition task: 1. An acoustic model that contains properties of each senome, a senome is a detector object to check for the presence of a phone in an utterance. 2. A phonetic dictionary that contains a mapping between words and the sequence of phones they are comprised of. 3. A language model, to restrict the search context when a given partial sequence of phones are identified. This greatly speeds up the computation. In a case where the language model is not an accurate representation of the given language the search may be wrongly restricted and may give undesired results.

#### 2.1.1 Advantages and Disadvantages of CMU sphinx

**2.1.1.1 Advantages** • State of art speech recognition algorithms for efficient speech recognition.

CMUSphinx tools are designed specifically for low-resource platforms

- Focus on practical application development and not on research
- Wide range of tools for many speech-recognition related purposes (keyword spotting, alignment, pronunciation evaluation)

#### 2.1.1.2 Disadvantages

- limited documentation and overview of data provided.

- acoustic build process takes many days.

## 2.2 Subtitle Generation using DeepSpeech

DeepSpeech is an open source embedded (offline, on-device) speech-to-text engine which can run in real time on devices ranging from a Raspberry Pi 4 to high power GPU servers. DeepSpeech: This is a Speech Recognition tool by Mozilla, which uses Long Short-Term Memory (LSTM) for speech recognition. DeepSpeech is trained on large RNN with multiple GPUs for thousands of hours. DeepSpeech is directly trained from data and does not require specialized components for speaker adaptation or noise filtering. The core of the system is the RNN layer, it is a bidirectional recurrent layer with two groups of hidden units, one for forward and one for backward recurrence. It consists of 2 convolutional layers, 5 bidirectional RNN layers and a fully connected layer. DeepSpeech is a character-level, deep recurrent neural network (RNN), which can be trained end-to-end using supervised learning. It extracts Mel Frequency Cepstral Coefficients (Imai, 1983) as features and directly outputs the transcription, without the need for forced alignment on the input or any external source of knowledge like a Grapheme to Phoneme (G2P) converter. The RNN layer uses LSTM cells, and the hidden fully connected layers use a ReLU activation function. The network outputs a matrix of character probabilities, i.e. for each time step the system gives a probability for each character in the alphabet, which represents the likelihood of that character corresponding to the audio. DeepSpeech comes with a pre-trained English model, but while Mozilla is collecting speech samples and is releasing training datasets in several languages, no official models other than English are provided. Users have reported on training models for French and Russian, but the resulting models do not seem to be available.

### 2.2.1 Advantages and Disadvantages of DeepSpeech

#### 2.2.1.1 Advantage

- provides an English speech recognition model.
- It is relatively easy to modify due to its Tensorflow implementation.

**2.2.1.2 Disadvantages** • DeepSpeech requires an exceedingly large amount of memory even when it is idle with only the model loaded. When this is compared to the memory usage of Sphinx which is just about 15 percent of DeepSpeech we begin leaning towards the Sphinx model and hope that with more training data and accounting for white noise while training we can produce better results.

## 2.3 Subtitle Generation using ASR API

there are different ASR tools available online (IBM Watson speech-to-text, google api, Microsoft api etc.). Automatic speech recognition (ASR) API for real-time speech that translates audio-to-text. Speech Recognition API captures your speech in real-time, transcribes it, and returns text. Transcribe a wide range of industry-specific words and phrases out of the box, without any pre-training. ASR is mainly used to Build responsive voice applications that act on partial recognition results as your customer speaks. There are many variations of deep learning architecture for ASR. Two commonly used approaches are: 1] A CNN (Convolutional Neural Network) plus RNN-based (Recurrent Neural Network) architecture that uses the CTC Loss algorithm to demarcate each character of the words in the speech. eg. Baidu's Deep Speech model. A regular convolutional network consisting of a few Residual CNN layers that process the input spectrogram images and output feature maps of those images. 2] An RNN-based sequence-to-sequence network that treats each 'slice' of the spectrogram as one element in a sequence eg. Google's Listen Attend Spell (LAS) model. A regular recurrent network consisting of a few Bidirectional LSTM layers that process the feature maps as a series of distinct timesteps or 'frames' that correspond to our desired sequence of output characters. (An LSTM is a very commonly used type of recurrent layer, whose

full form is Long Short Term Memory). In other words, it takes the feature maps which are a continuous representation of the audio, and converts them into a discrete representation.

## 2.3.1 Advantages and Disadvantages of ASR API

### 2.3.1.1 Advantages

- transforms complicated IVR menus to easy-to-use systems.
- ASR systems are key in tapping into this rich content and, in addition, can speed-up the transcription process as it does processing for real-time speech.
- improves security by eliminating agents from data processing.

### 2.3.1.2 Disadvantage

- Noisy environments, accents and multiple speakers may degrade results.
- regular voice recognition software can lack integration with other key services.
- It does not always work across all operating systems.

## Chapter 3

### Description

## 3.1 Analysis

In the previous sections, we have defined our problems clearly and surveyed the existing literature and approaches. Now we discuss how we analyzed this problem from our prospective.

### 3.1.1 Use Case Diagram

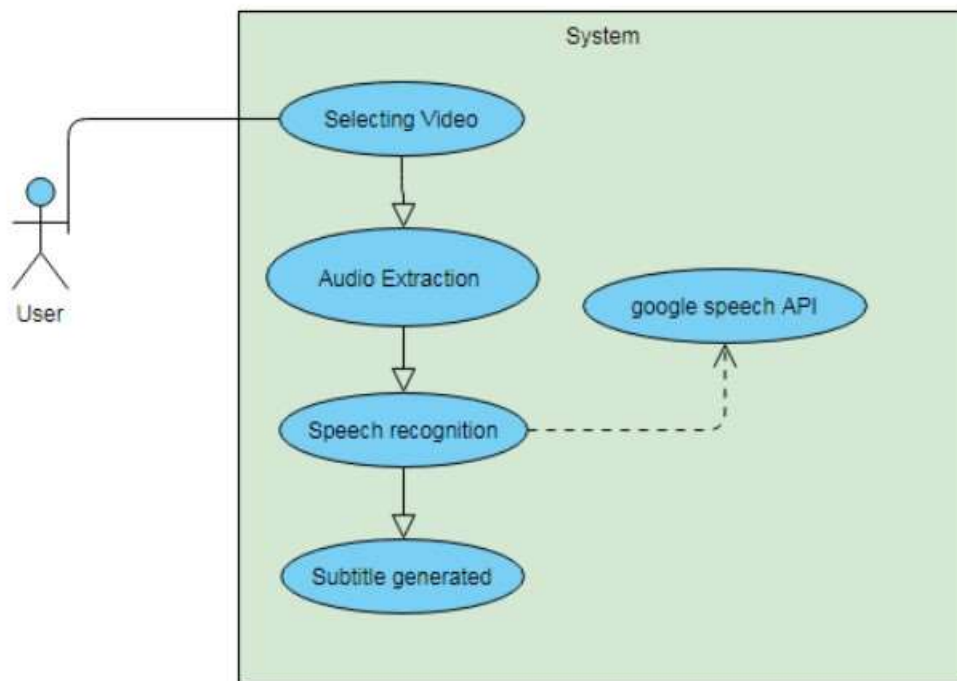


Figure 3.1: Use Case Diagram of our Proposed System

### 3.1.2 Feasibility Study

Here we discuss if we can achieve the planned aims of our project while staying within the constraints of our resources.

- Cost of data acquisition
  - How hard is it to acquire data?
  - How much data will be needed
- Cost of wrong Prediction
  - How frequently does the system need to be right to be useful?
- Availability of good published work about similar problems
  - Has the problem been reduced to practise
  - Is there sufficient Literature on the problem
- Computational Resources available for both training and inference
  - Will the model be deployed in a resource constrained environment

**3.1.2.1 Cost of Data Acquisition** • As we will be using available API which as ready made pre-trained dataset model, using available API gives the advantage of putting large number of advance trained model in lesser period of time.

- The basic rule of any good Machine Learning or Deep Learning project is the bigger the dataset, the more accurate the results will be. As stated above, Since API's ready made datasets available on the internet. At the same time having such large datasets needs a lot of monitoring and tailoring.

Nonetheless, we would like to augment our dataset so that even a simple algorithm would give us a good result.

**3.1.2.2 Cost of wrong Prediction** • For our subtitle generation project, we don't have a specific set of objective metrics in the conventional sense of right or wrong. The metrics we are using to judge our results are the uniqueness of the user interface for generating subtitle and the pleasing accuracy of the generated subtitle.

- Still from the perspective of probabilistic thinking, our algorithm needs to be right 7/10 times to fulfil the metric stated above.
- Also, we judge the performance of our model based on loss and accuracy of the model.

**3.1.2.3 Availability of good published work about similar problems** • Currently, it has been implemented to some extent, but the issue is the insufficiency of the research in some key areas related to this.

- As the research is still in its infancy, the associated literature in such a niche area is also limited. We have also tried to publish our own literature survey, hence enabling other who are interested in this area to research about it even more and contribute more to it.

**3.1.2.4 Computational Resources available for both training and inference** • The previous works and research in this area, all had an issue that they required great amount of data as well as computational resources. In our turn, we have tried to optimize the algorithm to work in a resource constrained computational environment as well as giving satisfactory results.

## 3.2 Design

### 3.2.1 Flowchart

In this section, we explain the high-level steps involved during the various phases of our project using flowcharts.

#### 3.2.1.1 Flowchart detailing the audio conversion process

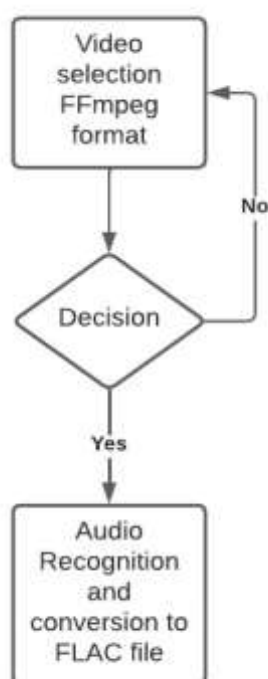


Figure 3.2: Flowchart explaining the audio extraction process

In simple terms, we ask the user to upload the following video in FFmpeg format, then the system checks if it the right format. In the next step, we send the video through the libraries where the audio is recognized from the video. Then we sent the audio to ASR model for subtitle generation.

### 3.2.1.2 Flowchart detailing the Subtitle generation process

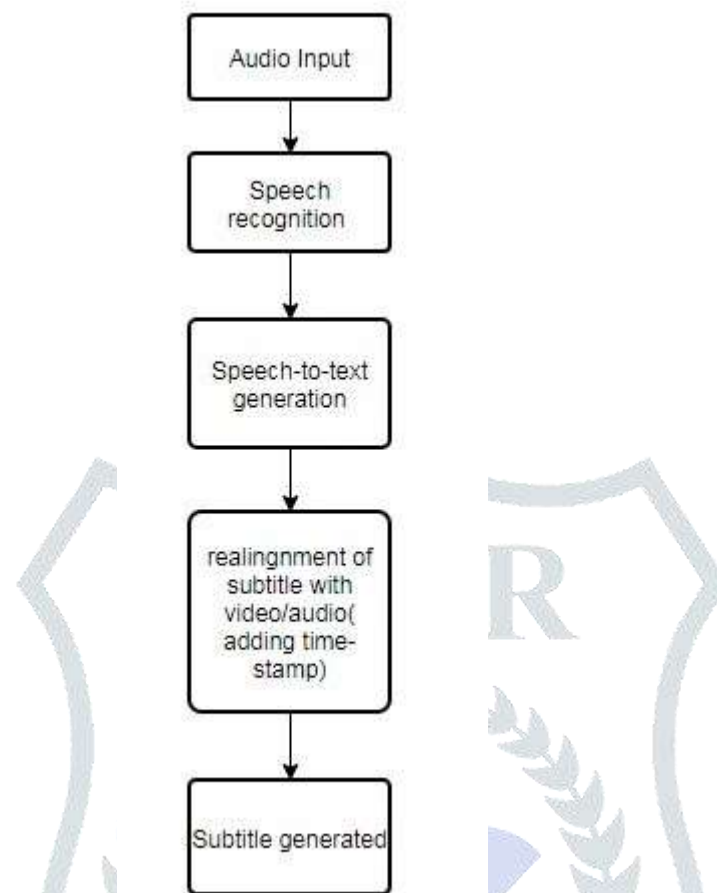


Figure 3.3: Flowchart explaining the algorithm for subtitling generation

In generation, our first step is to accept the audio file which then goes through the speech recognition model for audio recognition and then the recognized audio is converted from speech-to-text, after which realignment of the speech happens w.r.t the video/audio. These steps are followed to fulfill our goal of generating a subtitle in short period of time.

## 3.3 Details of Hardware and Software

The hardware with which we initially started was an AMD Radeon R5 M330 2GB GPU, but as the training of our model was very resource heavy, it was putting a strain on our hardware, hence we decided to use the GPU provided by Google Colaboratory, along with our hardware to reduce our training time and not put too much load on our hardware.

### 3.3.1 Google Colab

Colab is a free Jupyter notebook environment that runs entirely in the cloud. Most importantly, it does not require a setup and the notebooks that you create can be simultaneously edited by your team members - just the way you edit documents in Google Docs. Colab supports many popular machine learning libraries which can be easily loaded in your notebook.



### 3.3.2 PyQt

PyQt is a Python binding of the cross-platform GUI toolkit Qt, implemented as a Python plugin. PyQt is free software developed by the British firm Riverbank Computing. It is available under similar terms to Qt versions older than 4.5; this means a variety of licenses including GNU General Public License (GPL) and commercial license, but not the GNU Lesser General Public License (LGPL). PyQt supports Microsoft Windows as well as various flavours of UNIX, including Linux and MacOS (or Darwin).

### 3.3.3 Google API

Google APIs are application programming interfaces (APIs) developed by Google which allow communication with Google Services and their integration to other services. Examples of these include Search, Gmail, Translate or Google Maps. Third-party apps can use these APIs to take advantage of or extend the functionality of the existing services.

### 3.3.4 Jupyter Notebook

Project Jupyter is a nonprofit organization created to "develop open-source software, open standards, and services for interactive computing across dozens of programming languages". Spun off from IPython in 2014 by Fernando Pérez, Project Jupyter supports execution environments in several dozen languages.

### 3.3.5 FLAC

FLAC stands for Free Lossless Audio Codec, an audio format similar to MP3, but lossless, meaning that audio is compressed in FLAC without any loss in quality. This is similar to how Zip works, except with FLAC you will get much better compression because it is designed specifically for audio, and you can play back compressed FLAC files in your favorite player (or your car or home stereo, see supported devices) just like you would an MP3 file.

FLAC stands out as the fastest and most widely supported lossless audio codec, and the only one that at once is non-proprietary, is unencumbered by patents, has an open-source reference implementation, has a well documented format and API, and has several other independent implementations.

### 3.3.6 Autosub

Autosub is a utility for automatic speech recognition and subtitle generation. It takes a video or an audio file as input, performs voice activity detection to find speech regions, makes parallel requests to Google Web Speech API to generate transcriptions for those regions, (optionally) translates them to a different language, and finally saves the resulting subtitles to disk. It supports a variety of input and output languages and can currently produce subtitles in either the SRT format or simple JSON.

### 3.3.7 FFmpeg

FFmpeg is a free and open-source software project consisting of a suite of libraries and programs for handling video, audio, and other multimedia files and streams. At its core is the commandline ffmpeg tool itself, designed for processing of video and audio files.

# Chapter 4

## Implementation Methodology

### 4.1 Explanation of the Algorithm Used

Perceptual linear prediction (PLP) Perceptual linear prediction (PLP) technique combines the critical bands, intensity-to-loudness compression and equal loudness pre-emphasis in the extraction of relevant information from speech. It is rooted in the nonlinear bark scale and was initially intended for use in speech recognition tasks by eliminating the speaker dependent features. PLP gives a representation conforming to a smoothed short-term spectrum that has been equalized and compressed similar to the human hearing making it similar to the MFCC. In the PLP approach, several prominent features of hearing are replicated and the consequent auditory like spectrum of speech is approximated by an auto regressive all-pole model. PLP gives minimized resolution at high frequencies that signifies auditory filter bank based approach, yet gives the orthogonal outputs that are similar to the cepstral analysis. It uses linear predictions for spectral smoothing, hence, the name is perceptual linear prediction. PLP is a combination of both spectral analysis and linear prediction analysis.

Long Short-term Memory Recurrent Neural Networks (LSTM RNNs) Speech is a complex time-varying signal with complex correlations at a range of different timescales. Recurrent neural networks (RNNs) contain cyclic connections that make them a more powerful tool to model such sequence data than feed forward neural networks. RNNs have demonstrated great success in sequence labeling and prediction tasks such as handwriting recognition and language modeling. In acoustic modeling for speech recognition, however, where deep neural networks (DNNs) are the established state-of-the-art, recently RNNs have received little attention beyond small scale phone recognition tasks, notable exceptions being the work of Robinson, Graves, and Sak. DNNs can provide only limited temporal modeling by operating on a fixed-size sliding window of acoustic frames. They can only model the data within the window and are unsuited to handle different speaking rates and longer term dependencies. By contrast, recurrent neural networks contain cycles that feed the network activations from a previous time step as inputs to the network to influence predictions at the current time step. These activations are stored in the internal states of the network which can in principle hold long-term temporal contextual information. This mechanism allows RNNs to exploit a dynamically changing contextual window over the input sequence history rather than a static one as in the fixed-sized window used with feed-forward networks. In particular, the Long Short-Term Memory (LSTM) architecture, which overcomes some modeling weaknesses of RNNs, is conceptually attractive for the task of acoustic modeling. LSTM and conventional RNNs have been successfully applied to various sequence prediction and sequence labeling tasks. In language modeling, a conventional RNN has obtained significant reduction in perplexity over standard n-gram models and an LSTM RNN model has shown improvements over conventional RNN LMs. LSTM models have been shown to perform better than RNNs on learning context free and context-sensitive languages. Bidirectional LSTM (BLSTM) networks that operate on the input sequence in both directions to make a decision for the current input have been proposed for phonetic labeling of acoustic frames on the TIMIT speech database. For online and offline handwriting recognition, BLSTM networks used together with a Connectionist Temporal Classification (CTC) layer and trained from unsegmented sequence data, have been shown to outperform a state-of-the-art HiddenMarkov-Model (HMM) based system. Similar techniques with a deep BLSTM network have been proposed to perform grapheme-based speech recognition. BLSTM networks have also been proposed for phoneme prediction in a multi-stream framework for continuous conversational speech recognition. In terms of architectures, following the success of DNNs for acoustic modeling, a deep BLSTM RNN combined with a CTC output layer and an RNN transducer predicting phone sequences has been shown to reach state-of-the-art phone recognition accuracy on the TIMIT database. Deep BLSTM RNNs have recently been shown to perform better than

DNNs in the hybrid speech recognition approach. Using the hybrid approach, we have recently shown that an LSTM architecture with a recurrent projection layer outperforms DNNs and conventional RNNs for large vocabulary speech recognition, considering moderately-sized models trained on a single machine. In this paper, we explore LSTM RNN architectures for large-scale acoustic modeling using distributed training. We show that a two-layer deep LSTM RNN where each LSTM layer has a linear recurrent projection layer outperforms a strong baseline system using a deep feed-forward neural network having an order of magnitude more parameters.

The standard ASR systems delineate between phoneme recognition and word decoding. Before the emergence of deep learning, researchers often utilized other classification algorithms on highly specialized features such as MFCC in order to arrive at a distribution of possible phonemes for each frame. During the decoding phase, a Hidden Markov Model (HMM) with a pre-trained language model is used to find the most likely sequence of phones that can be mapped to output words. Earlier applications of deep learning in speech also separates the two tasks; there are many successful hybrid systems that take advantage of DNN's discriminative power for phone recognition but leave the decoding for the HMM. We are interested in extending the recent developments as done in that have produced state-of-the-art results with recurrent neural networks that can handle recognition and decoding simultaneously. With the advent of utilizing GPUs to train deep neural networks (DNNs), many DNN architectures have performed extremely well in a variety of machine learning problems. Even though typically there is little to no feature engineering in the process, these neural networks have managed to consistently beat benchmarks on various speech tasks. In fact, most of the state-of-the-art in automatic speech recognition are a result of DNN models. However, many DNN speech models, including the widely used Google speech API, use only densely connected layers. While such models have great learning capacity, they are also very prone to overfitting and it is also difficult for it to learn from features that have local correlations. Further, most such systems use a separate paradigm of model to perform decoding. With this insight, we'd like to combine the recent advancement in both CNN and RNN to produce an end-to-end speech recognition system using purely neural networks. The motivation to use CNN is inspired by the recent successes of convolutional neural networks (CNN) in many computer vision applications, where the input to the network is typically a two-dimensional matrix with very strong local correlations. While the reason for using RNN and CTC is to replace the HMM in order to truly achieve an end-to-end deep learning system.

## 4.2 Our Approach

### 4.2.1 Training

In this section, we will discuss how we curated the model and how we prepared it for using the API service for our project.

### 4.2.2 Data:

As we will be using available API which as ready made pre-trained dataset model, using available API gives the advantage of putting large number of advance trained model in lesser period of time. API being a third party application, we need to know the advancement done by the third party and implement the following.

Speech-to-Text On-Prem enables easy integration of Google speech recognition technologies into your on-premises solution. The STT On-Prem solution gives you full control over your infrastructure and protected speech data in order to meet data residency and compliance requirements. This best-in-class machine learning technology gives you access to next-generation speech recognition models that are more accurate, smaller in size, and require fewer computing resources to run than existing solutions

Key capabilities

High quality transcription it Applies Google's advanced deep learning neural network algorithms to automatic speech recognition. Efficient models Deploy efficiently with models that are less than 1 GB in

size and consume minimal resources. API compatible Full compatibility with the Speech-to-Text API and its client libraries. Supported languages Support your global user base with language supports in English, French, German, Spanish, Portuguese, Cantonese and Japanese.

### 4.2.3 Preparing the Data:

Now that we have decided on which API we have to use, we start with first deciding the input data formats and lengths and the requirements for the API functioning.

We specify the sample rate of your audio in the sample Rate Hertz field of the request configuration, and it must match the sample rate of the associated audio content or stream.

Sample rates between 8000 Hz and 48000 Hz are supported within Speech-to-Text.

We can specify the sample rate for a FLAC or WAV file in the file header instead of using the sample Rate Hertz field. A FLAC file must contain the sample rate in the FLAC header in order to be submitted to the Speech-to-Text API.

A pre-sourced data can ideally be asked capture audio using a sample rate of 16000 Hz. As Values lower than this may impair speech recognition accuracy, and higher levels have no appreciable effect on speech recognition quality.

However, if your audio data has already been recorded at an existing sample rate other than 16000 Hz, we do not re-sample your audio to 16000 Hz. Even for example, using sample rates of 8000 Hz, which may give accurate results but sometimes due to poor Hz the accuracy might differ.

#### Languages

Speech-to-Text's recognition engine supports a variety of languages and dialects. We have to specify the language (and national or regional dialect) of your audio within the request configuration's language Code field, using a BCP-47 identifier.

Example: Name Afrikaans (South Africa) af-ZA, Albanian (Albania) sq-AL

### 4.2.4 Creating the Model:

Finally we get to designing the automatic speech recognition model architecture.

Automatic Speech Recognition (ASR) Speech recognition is also known as automatic speech recognition or computer speech recognition which means understanding voice of the computer and performing any required task or the ability to match a voice against a provided or acquired vocabulary. ASR can be also used in bidirectional way i.e. Speech-to-text (STT). Speech recognition module of python gives the platform to convert spoken words into speech. This module is backed by Google speech API. It has its own pre-trained dataset and on that basis the conversion is taken place [14]. Flowchart -1: In this figure the flow of automatic subtitle generation is depicted in which the out is SRT file which is Speech Recognized Text. It has used the Autosub module of python. It keeps the Google dataset as the reference and the uttered word are gathered into the text file.

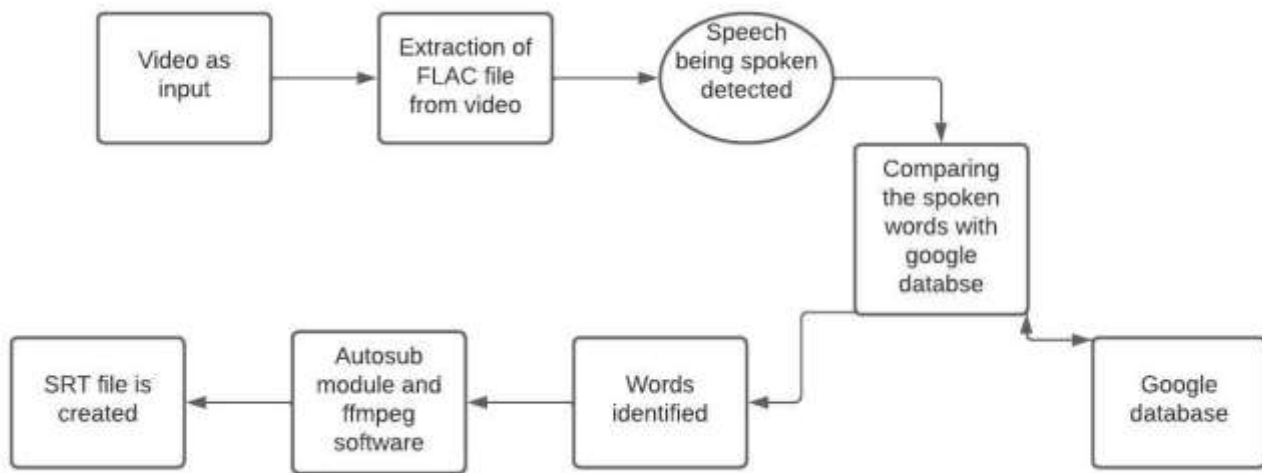


Figure 4.1: Flowchart explaining the model for subtitle generation

Autosub is a tool for automatic speech recognition and automatic subtitle generation. Video is taken as input from the FLAC file extracted which is Free Lossless Audio Codec file. It has very minute details of audio signal that's why it can't be played how we play .mp file. This file consist of MFCC features in the audio, those are compared Google pre-trained dataset and the actual word is estimated. Table-1 has explained it in very jest of the glance.

## Chapter 5

# Testing and Result

### 5.1 Testing

To evaluate the plausibility of our project's potential, we decided to conduct our own research through an different videos. Our intention was primarily to determine whether people who speak different languages are understanding the subtitle generated. And we also tested the videos of different quality to test the accuracy of the model we developed.

The metrics chosen to compare the speech recognition engines are mainly:

- 1) Word Error Rate.
- 2) Application Performance.

#### A. Word Error Rate

The metric we have decided to use help us quantitatively compare our various approaches at Speech Recognition is Word Error Rate (WER). This is the equivalent to the accuracy metric of traditional Machine Learning models where determining whether a prediction is true or not is easier. WER is Word Error Rate =  $(I + D + S)/N$  Here, I = No. of words inserted in a sentence D = No. of words deleted in a sentence S = No. of words substituted in a sentence N = Total no. of words in a sentence through our online research we have found that The average word error rate of DeepSpeech is 15.7% and the average word error rate of Sphinx is 20.4%. This result can be attributed to the superior architecture of DeepSpeech when compared with Pocket Sphinx namely the presence of a more robust inferencing mechanism. These results may vary depending on the video quality the the model developed by the developers. B. Application Performance

This performance test is conducted through the windows task manager which is an available windows application, and the data points are CPU load and system RAM usage of the application. These parameters are chosen because the application should be able to run on an end users machine which may not be high performing.

### 5.1.1 Evaluation of Subtitle generated by generated by our project

We took different videos of different quality and length and from those we have tried to get the average word error rate(WER) from which we can compare the result from sphinx and deepspeech models.

|     | Input Videos with varying audio quality in FLAC files           | Word error rate(WER) |
|-----|-----------------------------------------------------------------|----------------------|
| 1.  | Audio Quality 3500Hz                                            | 23%-25%              |
| 2.  | Audio Quality 4500Hz                                            | 19%-23%              |
| 3.  | Audio Quality 6000Hz                                            | 15%-18%              |
| 4.  | Audio Quality 7200Hz                                            | 12%-15%              |
| 5.  | Audio Quality 8000Hz+                                           | 12% and lower        |
| 6.  | Audio Quality with soft background noise                        | 12%-17%              |
| 7.  | Audio Quality with different ranging accent                     | 12%-15%              |
| 8.  | Audio Quality with loud background noise with lower audio sound | 25-28%               |
| 9.  | High Quality Audio                                              | 6%-8%                |
| 10. | High Quality Audio with zero background sound                   | 4%-5%                |

Figure 5.1: Results of the Subtitle generated by our project

Taking the result, we take the average of the resulted output we average to be anywhere around 13% and as mentioned above The average word error rate of DeepSpeech is 15.7% and the average word error rate of Sphinx is 20.4% so in comparison we see that the subtitle has slightly lower Word error rate.

### 5.1.2 Evaluation of our model

In this subsection we will compare the application processing the memory processing of the different models, these parameters include the CPU usage and the RAM usage by the application.

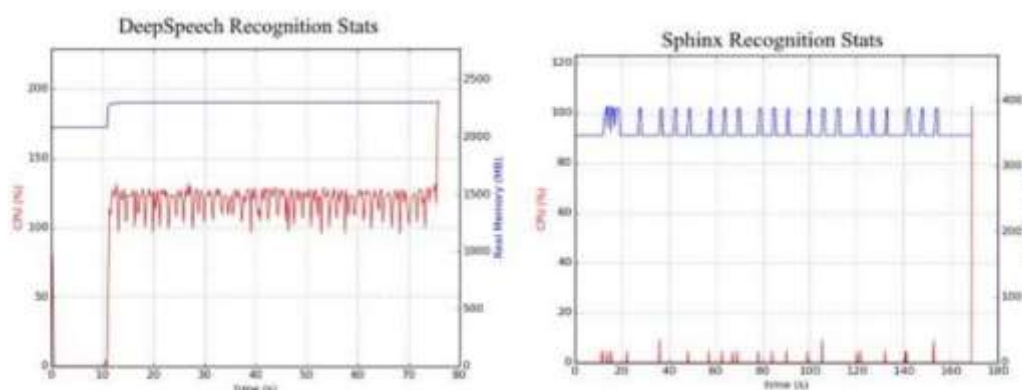


Figure 5.2: System resource utilization for Deepspeech and Sphinx model

above Figure shows the System Utilization of the two speech recognition models. What really is eye-opening and a concern for the overall feasibility of the application is the resource usage of the Speech Recognition System alone. DeepSpeech requires an exceedingly large amount of memory even when it is idle with only the model loaded. When this is compared to the memory usage of Sphinx which is just about 15% of DeepSpeech we begin leaning towards the Sphinx model and hope that with more training data and accounting for white noise while training we can produce better results.

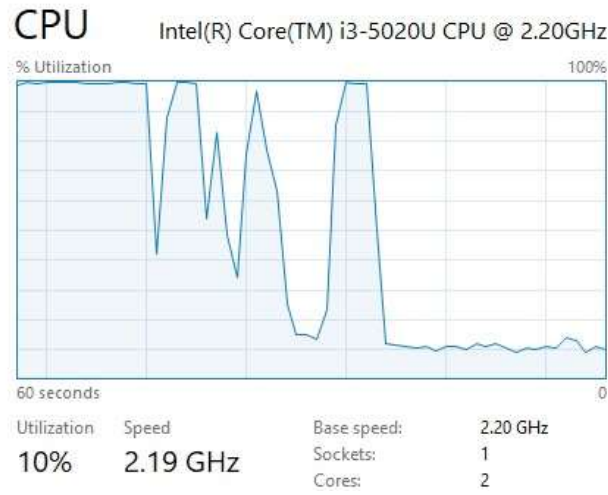


Figure 5.3: System resource utilization for our ASR API model

What we see is that the CPU consumption increases at a drastic rate for the conversion part but gradually decreases while transcribing the audio. In comparison to other models we see that the conversion part of our model is not that efficient to the computer but the transcribing part is highly efficient because it runs through online resources i.e. the API the resource of the computer drastically decreases and thus increasing the overall efficiency of the model. Due to this the time to create subtitle is faster than that of deepspeech and Sphinx.

## 5.2 Results

### 5.2.1 Screenshots of the project:

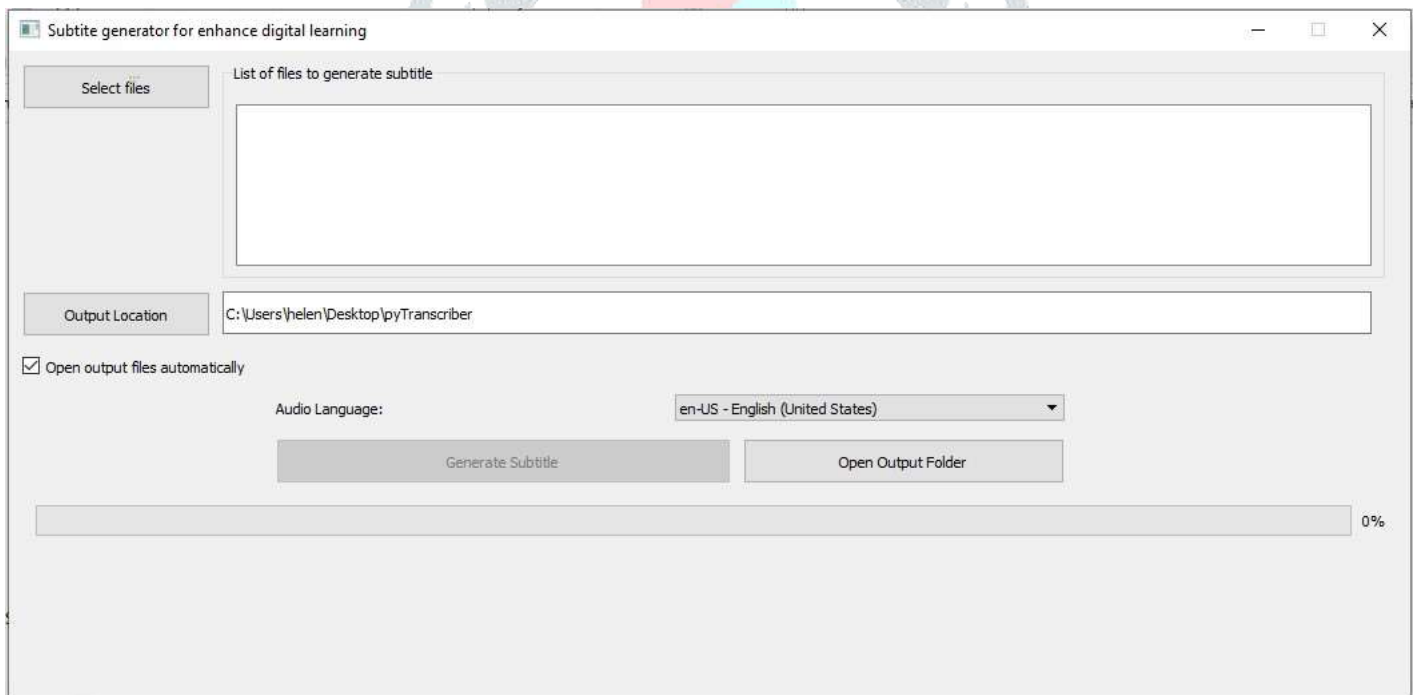


Figure 5.4: Opening the app and selecting the video

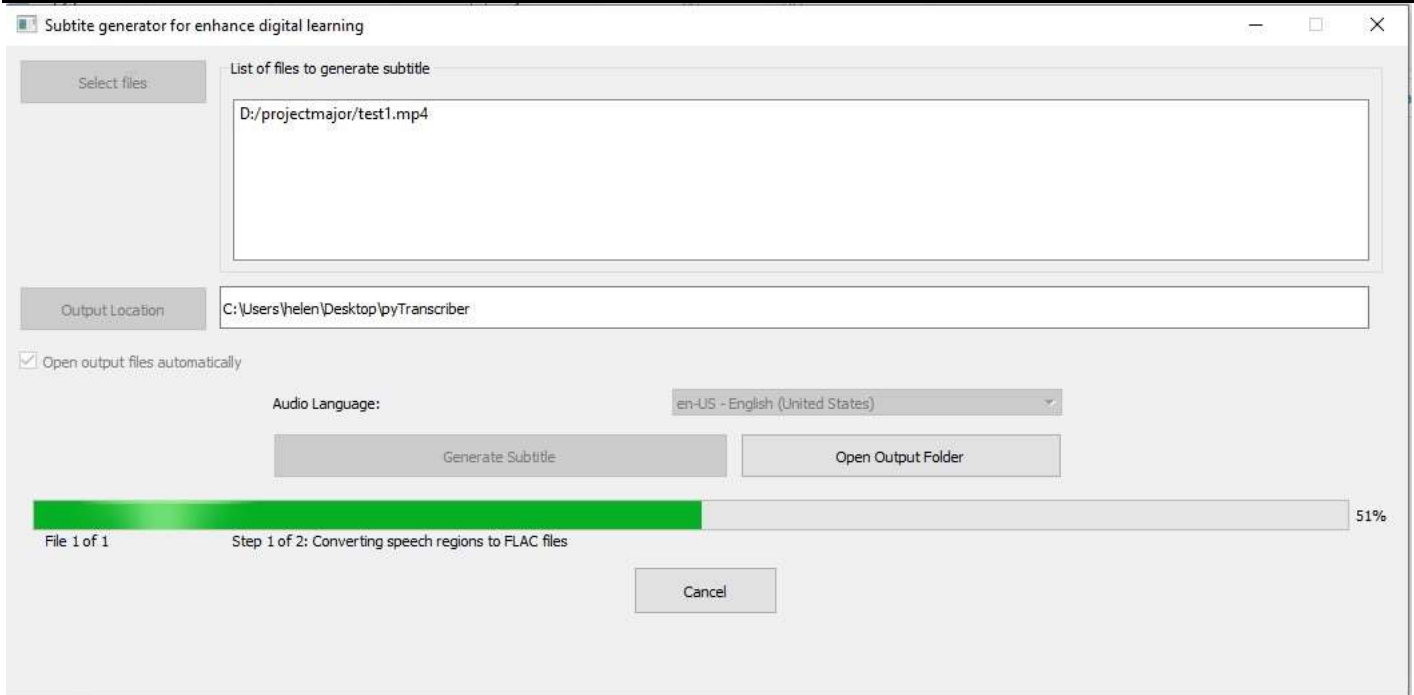


Figure 5.5: Starting the conversion of the video

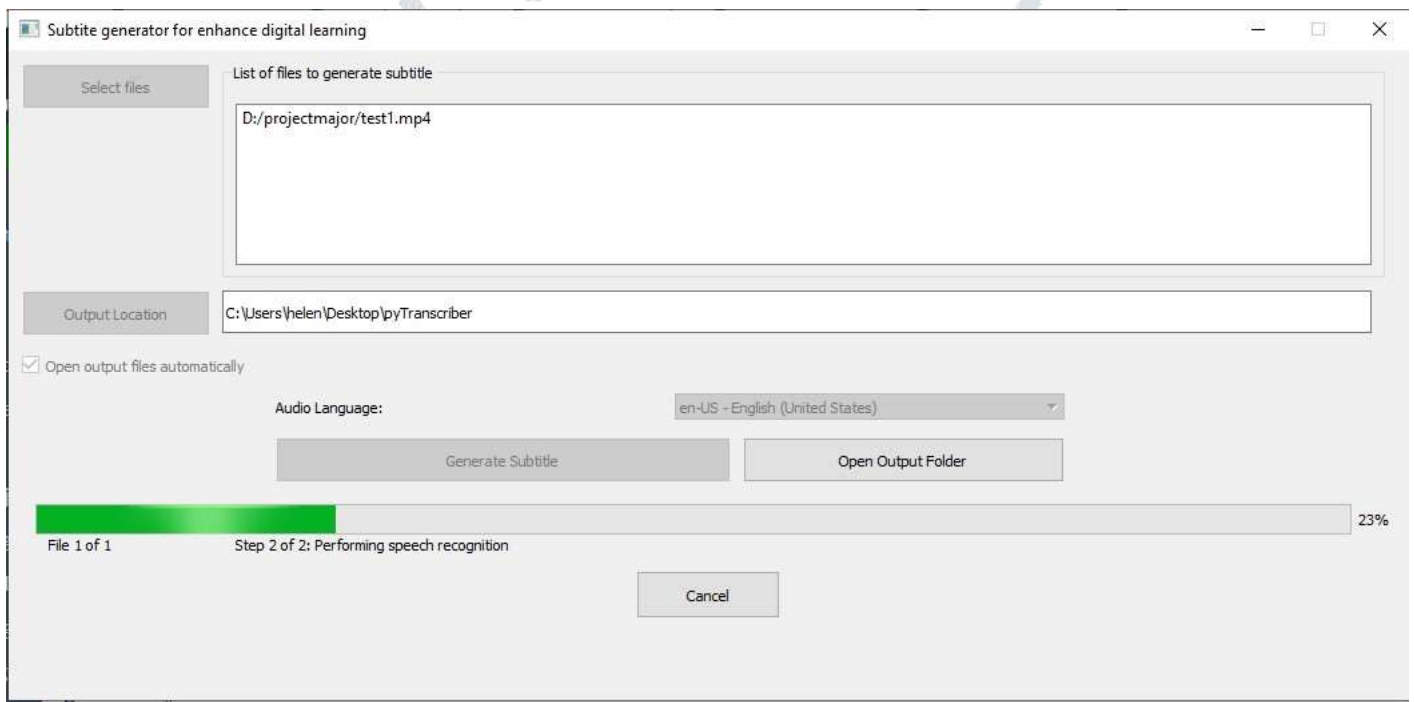
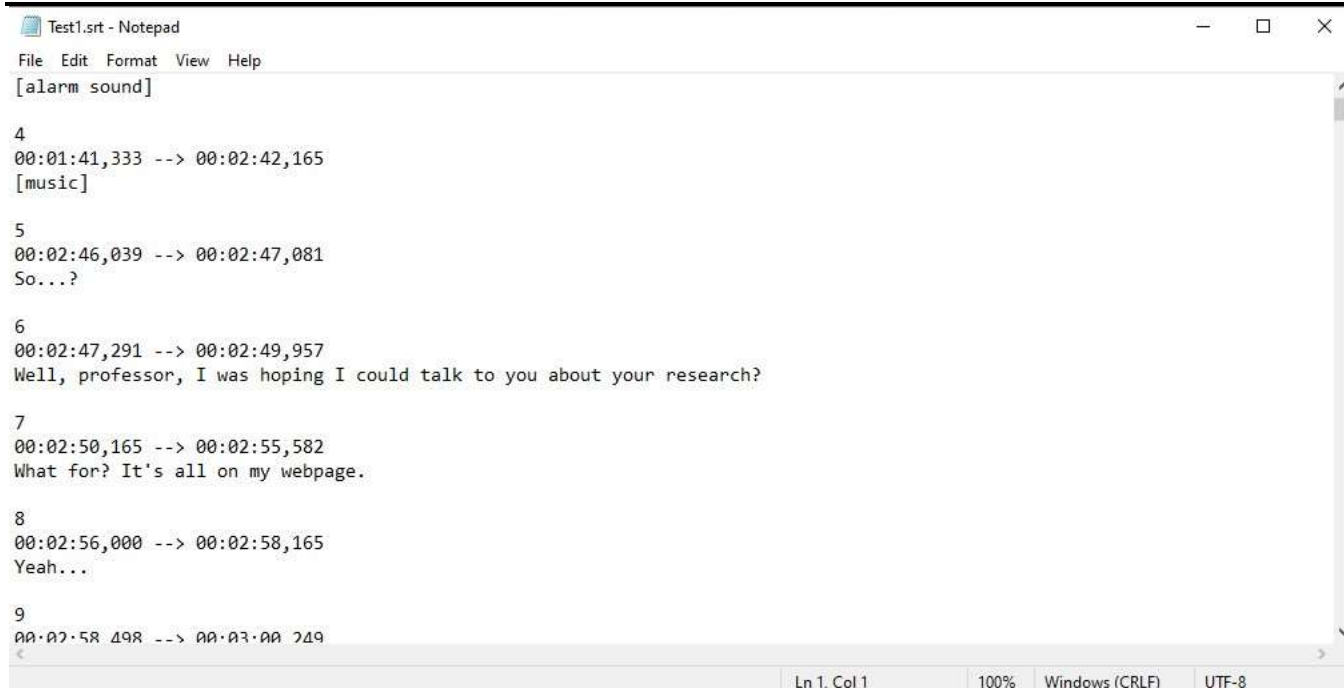


Figure 5.6: Starting of the transcribing of the video





```
Test1.srt - Notepad
File Edit Format View Help
[alarm sound]

4
00:01:41,333 --> 00:02:42,165
[music]

5
00:02:46,039 --> 00:02:47,081
So...?

6
00:02:47,291 --> 00:02:49,957
Well, professor, I was hoping I could talk to you about your research?

7
00:02:50,165 --> 00:02:55,582
What for? It's all on my webpage..

8
00:02:56,000 --> 00:02:58,165
Yeah...

9
00:02:58,498 --> 00:03:00,249

Ln 1, Col 1 100% Windows (CRLF) UTF-8
```

Figure 5.7: The final result

## Chapter 6

# Future Scope and Conclusion

### 6.1 Future Scope

In this thesis, we have described a model that is capable of producing subtitle based on the video and its language. However, there are some aspects or areas that can be improved upon.

- To provide a proper structure to our project where we can translate our subtitle, text file which can not only help people who are weak in that language but also people who are not familiar with that language.
- Although it is seems easy to use any media player for learning, we could try to develop our own media player.
- We could also try to increase the accuracy of the generated result if we can try to tweak the model and add a separate structure to increase the quality of the audio which in turn increases the accuracy of the result.

### 6.2 Conclusion

Video captions, also known as same-language subtitles, benefit everyone who watches videos (children, adolescents, college students, and adults). More than 100 empirical studies document that captioning a video improves comprehension of, attention to, and memory for the video. Captions are particularly beneficial for persons watching videos in their non-native language, for children and adults learning to read, and for persons who are deaf or hard of hearing. However, despite U.S. laws, which require captioning in most workplace and educational contexts, many video audiences and video creators are naïve about the legal mandate to caption, much less the empirical benefit of captions.

Subtitles make videos more accessible to a wider audience, including foreign-language speakers, hard-of-hearing individuals, and anyone who can't watch a video with sound. They also help improve engagement and boost SEO for content producers. So generating a subtitle successfully can solve different issues and also enhance the concept of digital learning.