

FYP Final Report
AUTOMATED DUBBING SYSTEM

Final Year Project Report
by

AMNA ABID 263387
ZAINAB BINTE IFTHIKHAR 242240

In Partial Fulfillment
Of the Requirements for the degree
Bachelor of Engineering in Software Engineering (BESE)

School of Electrical Engineering and Computer Science
National University of Sciences and Technology
Islamabad, Pakistan
(2022)

DECLARATION

We hereby declare that this project report entitled “Automated Dubbing System” submitted to “SEECS,” is a record of an original work done by us under the guidance of Supervisor “Dr. Rafia Mumtaz” and that no part has been plagiarized without citations. Also, this project work is submitted in the partial fulfillment of the requirements for the degree of Bachelor of Software Engineering.

Team Members

Signature

Amna Abid

amna abid

Zainab Binte Ifthikhar

zainab iftikhar

Supervisor:

Signature

Dr. Rafia Mumtaz

Dr. Ali Tahir

Date:

20th May 2022

Place:

NUST,SEECS

Sector H-12

Islamabad

DEDICATION

We as a team dedicate this project to our Family, Friends, Mentors and SEECs faculty. Without their support, this venture would not have been a successful one. We hope the extracts of this project will bring forth more development and progress.

Inshahallah

ACKNOWLEDGEMENTS

This project would never have been possible without all the guidance and help we received from our supervisor Dr. Rafia Mumtaz, and our co-advisor Dr. Ali Tahir. We are extremely grateful to our aforementioned supervisors for their valued assistance and mentorship in every capacity that was required of them. We would also like to extend our gratitude to our peers who helped us in dataset gathering and requirement gathering discourse. Special thanks to our Lab Engineers and other faculty members who helped us resolve our coding issues and assisted us in attaining a fruitful direction for our venture.

TABLE OF CONTENTS

ABSTRACT..... 7

INTRODUCTION..... 9

LITERATURE REVIEW 12

PROBLEM DEFINITION 15

METHODOLOGY..... 17

DETAILED DESIGN AND ARCHITECTURE 22

IMPLEMENTATION AND TESTING..... 35

RESULTS AND DISCUSSION..... 36

CONCLUSION AND FUTURE WORK 37

LIST OF FIGURES

<i>Figure. 1</i>	10
<i>Figure. 2</i>	10
<i>Figure. 3</i>	11
<i>Figure 4. Project Methodology</i>	18
<i>Figure. 5</i>	19
<i>Figure. 6</i>	19
<i>Figure 7. Audio File saved as a Blob</i>	20
<i>Figure 8. Urdu not listed among supported languages for Google Text-to-Speech</i>	21
<i>Figure 9. Dubbed Audio superimposed with Original Video</i>	21
<i>Figure 10. Upload Section of Web App</i>	24
<i>Figure 11. Upload Section with API Model selected</i>	25
<i>Figure 12. Web App with Custom Model selected</i>	26
<i>Figure 13. Flask Working</i>	27
<i>Figure 14. Work Flow of ngrok</i>	28
<i>Figure 15. Output Folder</i>	29
<i>Figure 16. Result Folder</i>	29
<i>Figure 17. System Architecture</i>	30

ABSTRACT

About 60% of the content online is available in English language, may it be in form of articles, videos, or blogs. Most of the educational content seen on learning websites like Udemy, Coursera and YouTube, is in English. However, less than 8% of Pakistan's population can speak English as a first language. This means that other than entertainment options, most of Pakistan's population has very little access to educational content posted online. Considering the wealth of knowledge and learning that is obtained from the internet all over the world, the statistics show that there is an eminent need for a way to make this content accessible to Pakistan's population.

Since most web content consumed today is in the form of videos, a considerable work has been directed to make them accessible to a wider population of the world. Examples of this can be seen in form of caption generation, translation and speech synthesis technologies introduced by Google and employed in their products such as YouTube and search engines. Though these technologies have come a long way in trying to reduce the barrier between online English content and different language consumers, there is still a lack of support available for our national language Urdu.

We propose a system that can use existing and custom developed tools to convert educational videos into Urdu language that can then be understood by the majority of our population. This system will provide content creators a way to increase their target audience and will also allow consumers a chance to watch videos in their own

language. Our proposed solution puts forward two methodologies: (1) A dubbing system developed through existing APIs and libraries, and (2) A system based on custom trained translation model fine-tuned for translation of English into Urdu. Both configurations of the system will be made accessible through a simple and easy to use website called ‘DubInc.’ This will allow users to easily dub their videos without any complexity.

INTRODUCTION

Dubbing of a video involves technologies that allow a video in one language, often referred to as source language, to be converted into a video in the desired or target language. The process of dubbing involves the integration of many small subsystems for transcribing the video, converting the source text into target language and synthesizing speech from the translated text. The involvement of so many technologies means that with advancement in these subsystems, the whole dubbing system has a chance to improve as long as these technologies are improving. Some applications of these technologies can already be seen in form of mobile assistants like Siri and automatic captioning on YouTube. Application of these technologies to provide systems for the consumption of content in different languages not only increase the target audience of different content but also allows for greater consumer satisfaction. On average, about 65% of people prefer the content they consume to be in their native language even if the quality is poor [1]. Therefore, the need for dubbing system is evident.

Fig. 1 shows how a basic dubbing system works. First, the original video in the source language is used to extract audio. The extracted audio is then transcribed which results in a script of the original video in the source language. This text is then translated into the desired language. Passing through this module results in text in the target language. This translated text is then utilized to synthesize speech in the target language according to the desired phonemes and speech style requirements. This step results in an audio file in the target language. Now, as the last step in dubbing, the audio is superimposed on the original video in place of the source audio to result in a dubbed video in the required target language.

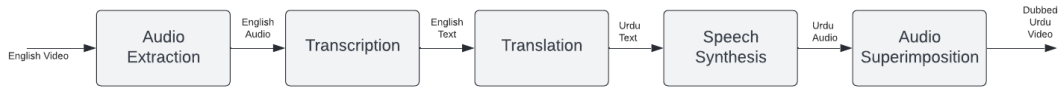


Figure. 1

The dubbing system displayed by Fig.1 can be implemented in many ways by combining a variety of technologies already available or through custom systems. For the purpose of this FYP, a two-pronged approach to the problem was considered to carry out a comparative analysis. As a result, two systems were developed:

1. An API Model
2. A Custom Model

Fig. 2 shows the high-level implementation of the API Model. For developing this system, a variety of Google APIs were used along with some python libraries. This decision was taken after thorough analysis of the existent APIs in market for this purpose and their respective results as well as required resources. This resulted in a compilation of existent technologies to develop an end-to-end system that can receive a video in English language and give an Urdu dubbed video in response. A system based on APIs can be implemented far more quickly and with more ease as compared to custom systems. However, this ease comes at a cost of rigidity in system modules that can prevent such systems to be fine-tuned to fulfil certain requirements.

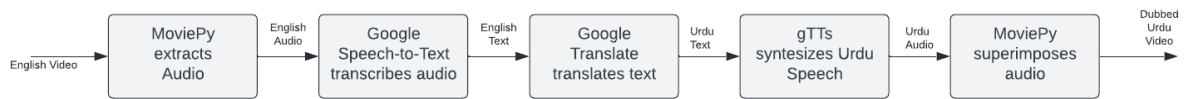


Figure. 2

For the second system, i.e., the Custom Model, an approach using RNNs, and statistical models was considered. After some experimenting, we decided to implement a system with a specialized translation module developed by using RNN and LSTM. Fig. 3 shows that this system follows the same route as the API model with the exception that

translation is now achieved through a custom-made model instead of an API. As we wanted our system to be ‘*content-based*,’ therefore we decided to focus on the quality of translation between source and target language to convey the maximum information as a result of dubbing.



Figure. 3

To develop a flexible translation module that could handle a variety of translation scenarios, we chose an encoder-decoder LSTM. This architecture can prove very useful in translation of languages as sometimes the input (source) sentence can have a different length than an output (translated) sentence and the encoder-decoder model can handle that variation gracefully. The basic architecture of the encoder-decoder model as observed in Fig. 4 shows that the model consists of three parts:

1. **Encoder:** To process a single part of the input sentence and propagate this information forward
2. **Hidden States:** The final result of the encoder module containing information about the entire input sequence
3. **Decoder:** To predict the translated counterpart of the source words given the input sentence

In our paper, we will use both these models, i.e., the API model and custom LSTM model to develop an end-to-end system for the dubbing of English videos into Urdu. As there is significant work done already in the field of machine translation, we will utilize all the lessons learned to develop a system fine-tuned for the Urdu language. We will explore different architectures and modifications to the system to achieve the best result so that the system can one day be utilized to provide access of English video to the Urdu speaking majority of Pakistan that consists of around 120 million individuals (thirty million native speakers and ninety-four million people who speak Urdu as a second language).

LITERATURE REVIEW

Work on automatic dubbing systems has been carried out for quite a few years now making progress with each attempt. As a result, the systems have become better and more suitable to be used commercially as far as accuracy is concerned. One such example can be seen in the form of the lip-sync model proposed and implemented by [2]. The system focuses on creation of a system that can tackle the problems of the alignment of dubbed speech and the input video. As opposed to other research work in the domain, the system put forward in this work aims to produce an end-to-end system that can hope to replace the manual dubbing and subtitling system in use today.

The proposed system consists of two major modules namely, subtitling and audio dubbing. Subtitling of a video in a foreign language can severely impact the enjoyment and presence of the viewer as exhibited by [3]. This is because none of these methods can retain the spatial and gesture information of the video speakers with the audio being produced. This can cause misalignment, loss of pertinent information shown through gestures and expressions. And thus, the resultant videos are not able to keep the attention of viewers long because of the requirement of extra effort to piece information together [4]. The automated dubbing system first extracts audio from the provided video. The audio is then transcribed to get text from it. The text obtained is translated from the source language to the target language. Afterward comes the step of text-to-speech (TTS). However, this paper considers the essential problem that the audio synthesized from the translated text does not cover the same time as its source language counterpart, does creating the misalignment targeted by the work. The study deals with this problem by proposing a lip-sync model that adjusts the output translated text according to the prosody of the input sentences

Other efforts to create end-to-end systems to cater the dubbing process include systems focusing on focusing on the commercial application in the movie industry for the task of Automated Dialogue Replacement (ADR) such as [5]. This system differs as compared to the one presented in [2] its area of application as well as the methodology utilized. Unlike the system proposed by [2] that catered the general population in an effort to make English learning content available to all, the system put forward by [5] is fine tuned for the task of dubbing to be utilized by the film making industry. In that respect, it tackles two main issues: (1) achieving synchronization and alignment between the translated text and the video segments (2) maintaining the same information in translated text as was contained in the source text.

In order to implement the system as designed and to improve upon its efficiency at the same time, research was done for this problem statement in the field of Machine Learning. Understanding existing systems is important, just as understanding the models behind existing systems is.

Machine Learning has evolved rapidly over the last few decades. While evolving, it has also been experimented with and is still in its experimental phase as of now. Various models and algorithms, as a result of this, have started to outperform traditional and conventional systems.

One such example is of neural networks. Neural networks have wreaked havoc in the field of machine learning. It has evolved a field of its own, deep learning. The concept of teaching a machine how to identify patterns and learn on its own has done a phenomenal job in creating better working and more accurate systems.

similar to this, neural networks are now being used to develop an end-to-end system where the machine performs machine translation on the given data and outputs it. This output is then used to create a voice through speech synthesis.

For instance, in [6], researchers used a modular approach to allow their users to change the composition of their modular components as they pleased to. They used deep learning models to create an end-to-end system that performs this task in an automated manner but in a specific target language.

PROBLEM DEFINITION

Most of the entertainment and educational content consumed today consists of videos. About 720,000 hours of video content is uploaded on YouTube alone with the video-streaming giant getting two billion active user every month. Accessibility of internet all over the world means that this content is now composed of videos not just in a single language but multiple languages with English, American, Hindi, Portuguese and Thai being the top five languages used. Out of these top five languages, two are different dialects of English, gaining more than 1300 billion views.

However, the people in Pakistan who can understand and speak English well comprise of only 8% of the population at best. Thus, most of the content posted online is inaccessible by the masses and poses barriers towards access to different forms of education and entertainment. This puts forward the issue of making the huge amount of online video content available to the people in Pakistan so that there are fewer barriers in learning and understanding.

As far as dubbing and favor towards this medium of content consumption is concerned, the biggest streaming services including Hulu, Disney, and most notably Netflix that has been actively investing to bring back dubbing and increase viewers on the platforms. Netflix also reports a 120% increase in the consumption of its dubbed content. The streaming giant has, in turn, increased its efforts to provide content in more languages through collaboration with more than 125 dubbing facilities regularly [6]. YouTube has also not lagged, although the effort on the platform's front is from its content creators rather than the platform itself. YouTubers have taken it upon themselves to hire voice actors and other resources to dub their content and increase

viewership across the world. However, support for Urdu from both content creators and streaming platforms is little to none, thus sparking the need for a system that can adequately tackle the challenge and make global content accessible to Pakistan's Urdu speaking population.

METHODOLOGY

Videos have become the most widely consumed form of content throughout the world due to better understanding and engagement. However, most of the video content on the internet is in English which makes it difficult to be as engaging and understandable to the general population of Pakistan as it intends. Therefore, there is a need to fashion a system that can provide the general population a way to consume English content, and more specifically, English videos, with more ease. Our team proposes this solution in the form of a dubbing system that can dub English videos into Urdu language, thus making these videos available and comprehensible to the Urdu speaking population.

Our proposed solution consisted of two ways for tackling the problem. The first milestone was to develop a dubbing system from the already existent technologies including APIs and pre-existing libraries. The second milestone was set to achieve the development of a custom translation module that could increase the accuracy of translation in the dubbed videos provided by our system so as to maintain content credibility.

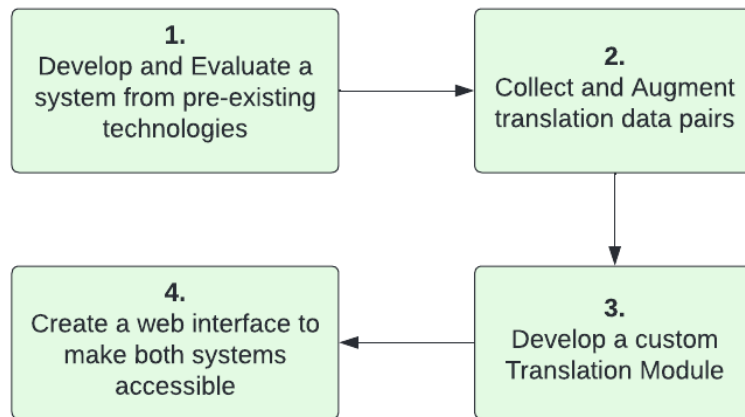


Figure 4. Project Methodology

Phase 1: API System

Based on our preliminary research, we decided to approach towards the development of the dubbing system with widely available components first and foremost. This was done through the help of Application Programming Interfaces or APIs, in other words. APIs allow us to communicate with certain software applications through a simple interface. For the purpose of developing our dubbing system, we decided to use Google Cloud APIs. In this reference, the APIs used were as follows:

1. Google Speech-to-Text
2. Google Translate
3. Google Text-to-Speech

Google Cloud services including storage buckets and APIs can easily be used after some initial steps of setting up a service account to access these services. Therefore, to be able to use these APIs, the approach was as follows:

- Create a project on Google Cloud
- Create a service account
- Generate a JSON access key

This key was then utilized to set up the system environment so that the APIs could be accessed easily.

To set up the API system, we used Google Colab which is built upon Jupyter notebooks and provides support for Python as well as Flask. In order to use Google APIs as well as storage, the Colab environment variable called

`GOOGLE_APPLICATION_CREDENTIALS` was set equal to the path of the JSON key that provided access to the service account that we set up earlier. For our system, we stored this file in Google Drive, the path for which we set equal to the variable as shown in Fig 5.

```
[ ] import os
    os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = '/content/drive/MyDrive/Colab Notebooks/FYP Frontend/fyp-api-key.json'
```

Figure. 5

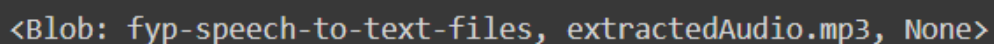
To extract the audio from the uploaded video, we used Python library called MoviePy that helped us extract the audio from provided video files in the form of an mp3 file. This audio is written into an mp3 file and saved as shown by Fig 6. By default, this file is saved to the temporary Colab environment. However, since the file was required for further processing in the system, the loss of file due to internet issues was prevented by saving it in a folder in Google Drive.

```
↳ [MoviePy] Writing audio in extractedAudio.mp3
100%|██████████| 1328/1328 [00:01<00:00, 714.61it/s]
[MoviePy] Done.
```

Figure. 6

The extracted audio is then transcribed through Google Speech-to-Text API. By default, a user can transcribe up to 1 minute of a video with Google-Speech-to-Text. However,

for longer files, google storage buckets are required. Since there is no guarantee that our system will only need to transcribe videos of maximum 1 minute, we implemented a function to upload the extracted audio file on Google Storage buckets as a blob as Fig 7 shows. The transcription function then used this storage bucket to get the audio file to be transcribed for asynchronous transcription. This function returns a JSON format result that contains both the transcriptions for each sentence as well as the confidence for the relevant transcription. The transcription was extracted from this result to be used in further functions.



```
<Blob: fyp-speech-to-text-files, extractedAudio.mp3, None>
```

Figure 7. Audio File saved as a Blob

To translate this text, Google Translate API was used. For this purpose, 'utf-8' encoding was used along with Urdu as the target language. This function resulted in translated text. Now, as the next step, this text needed to be converted into audio to be used for dubbing. For speech synthesis, we decided to use Google Text-to-Speech provided by the Google Core. However, when looked into the support for languages provided by Google Text-to-Speech, Urdu was not listed among supported languages as shown in Fig 8. It was also observed that the API did not work when given Urdu language code 'ur.' To navigate this obstacle, python library called *gTTs* was used that is built upon a previous version of Google Text-to-Speech that supports Urdu language.

Turkish (Turkey)	WaveNet	tr-TR	tr-TR-Wavenet-D	FEMALE	
Turkish (Turkey)	WaveNet	tr-TR	tr-TR-Wavenet-E	MALE	
Ukrainian (Ukraine)	Standard	uk-UA	uk-UA-Standard-A	FEMALE	
Ukrainian (Ukraine)	WaveNet	uk-UA	uk-UA-Wavenet-A	FEMALE	
Vietnamese (Vietnam)	Standard	vi-VN	vi-VN-Standard-A	FEMALE	
Vietnamese (Vietnam)	Standard	vi-VN	vi-VN-Standard-B	MALE	

Figure 8. Urdu not listed among supported languages for Google Text-to-Speech

To finish the process of video dubbing, the synthesized speech needed to be combined with the video in order to produce a dubbed video output. For this purpose, MoviePy was used to superimpose the dubbed audio with the original video as uploaded by the system user as Fig 9 shows. The final video was then saved to Google Drive to be accessed by the final web portal for viewing and download by the user.

```

/content/sample_vid_onemin.mp4
[MoviePy] >>>> Building video resultVid.mp4
[MoviePy] Writing audio in resultVidTEMP_MPY_wvf_snd.mp3
100%|██████████| 1841/1841 [00:01<00:00, 1033.15it/s][MoviePy] Done.
[MoviePy] Writing video resultVid.mp4

100%|██████████| 1806/1807 [00:04<00:00, 415.94it/s]
[MoviePy] Done.
[MoviePy] >>>> Video ready: resultVid.mp4

<moviepy.video.io.VideoFileClip.VideoFileClip at 0x7f0b4a20cb50>

```

Figure 9. Dubbed Audio superimposed with Original Video

DETAILED DESIGN AND ARCHITECTURE

a. SYSTEM ARCHIECTURE

Our Automated Dubbing System will comprise of multiple parts for proper functioning. However, at a top level, three modules can be considered: the API Model, the Custom Model, and the Web Interface for interacting with the system. The primary goal of the project is to provide a platform that users can easily use to dub any English video into Urdu. This can help users not only consume videos in their own language but be able to interact with the content in a lot better and comfortable manner. This system can be utilized to dub videos of any nature, may it be entertainment, news, or education.

The system will be developed as a web application so that users can easily navigate through the system to dub their videos. For this purpose, the interface will also be kept fairly minimal to promote quick access and ease of use. Our system differs from those in the market in the fact that at present, there are very few end-to-end systems ensuring complete dubbing of videos. And the ones that are available either do not support English to Urdu dubbing or give poor results, making it inconvenient.

Our project will utilize two approaches to solve the lack of end-to-end dubbing systems. One approach will be to utilize already existing components and tools to fashion a complete system that can dub English videos into Urdu without hassle. The other approach will be to develop a fine-tuned translation model based on deep learning to improve the performance of the API system towards a

system that can provide efficient and smooth results. Both of these modules will be made usable through a web interface where a user can upload the video to dub. The interface will also provide the user option of wither dubbing with the API system or the Custom system. The output of the system will be the original video dubbed in Urdu that can be viewed by the user on their system and also be downloaded.

i. Architecture Design Approach

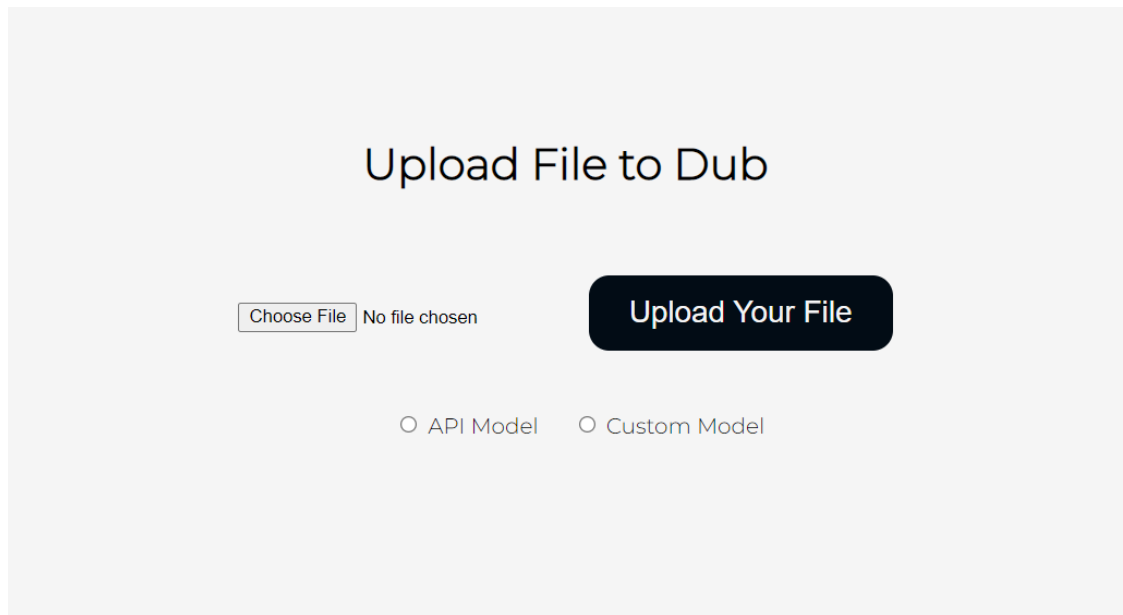
The high-level components of the system include the web app, the API Model, and the Custom Model. Their interaction is explained as follows:

- **Web Application**

The dubbing system is hosted on a web app called DubInc that is connected to two dubbing models at the backend. The user uploads a video from local computer through the upload file button. When the video is uploaded to the system, it generates a request for dubbing at the backend depending on user's choice of model: API or Custom.

- **Front-End**

The web app uses an upload button to make it easier for user to choose a video file from the local system for dubbing. The button can be seen in Fig 10.



Upload File to Dub

Choose File No file chosen

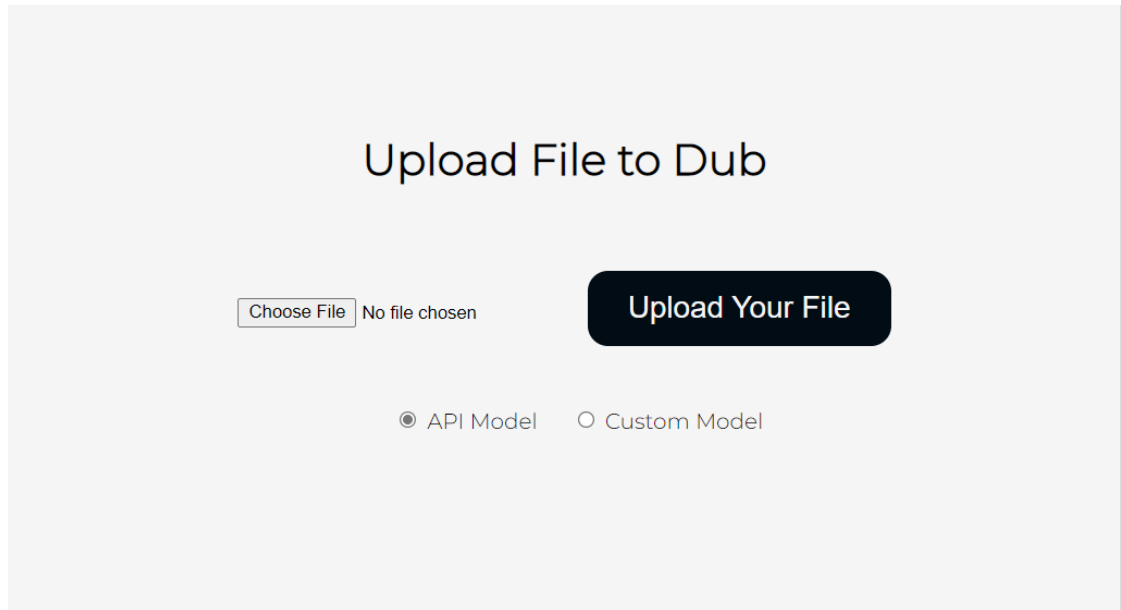
Upload Your File

☐ API Model ☐ Custom Model

Figure 10. Upload Section of Web App

- **API Model**

The API model integrated with the web app through python framework called Flask. The user can choose the API model by clicking the radio button next to API Model option on the web app when uploading the file for dubbing. This option can be seen through the Fig 11.



Upload File to Dub

Choose File No file chosen

Upload Your File

☒ API Model ☐ Custom Model

Figure 11. Upload Section with API Model selected

- **Custom Model**

The custom model is integrated with the web app in a similar fashion to the API Model. Flask is used for integration while user can choose to dub their file through the custom model by clicking the radio button right next to the Custom Model option on the front end. This option is displayed in Fig 12.

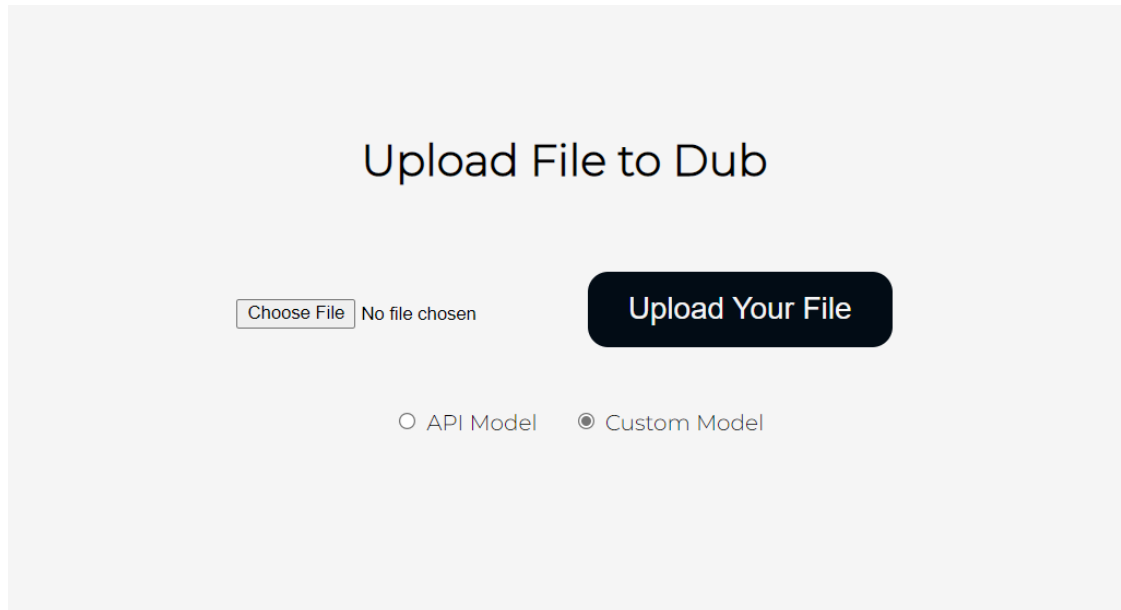


Figure 12. Web App with Custom Model selected

ii. Subsystem Architecture

1. Web Application

When you look at the system, the first observable thing is the frontend or the website. We decided to implement a web app called 'DubInc' where the user could upload an English video to dub it. The video file is saved in the backend of the system. Then, the user can select the type of system to dub the video. Here, the user will have the option to select one of two models: the API model or the Custom model.

As the front end of the system is integrated with the models in the backend through Python's Flask library, the form used for uploading the video and submitting it for dubbing is also done through Flask's WTForms library that allows for easy rendering and use of forms in Flask like the one used in our system. When the user

selects a model for dubbing, and then clicks on the button to dub it, a POST request is sent, along with the video file, to the relevant model at the backend to process the video. When the dubbed video is generated, it is fetched to the web app for display through an HTTP GET request.

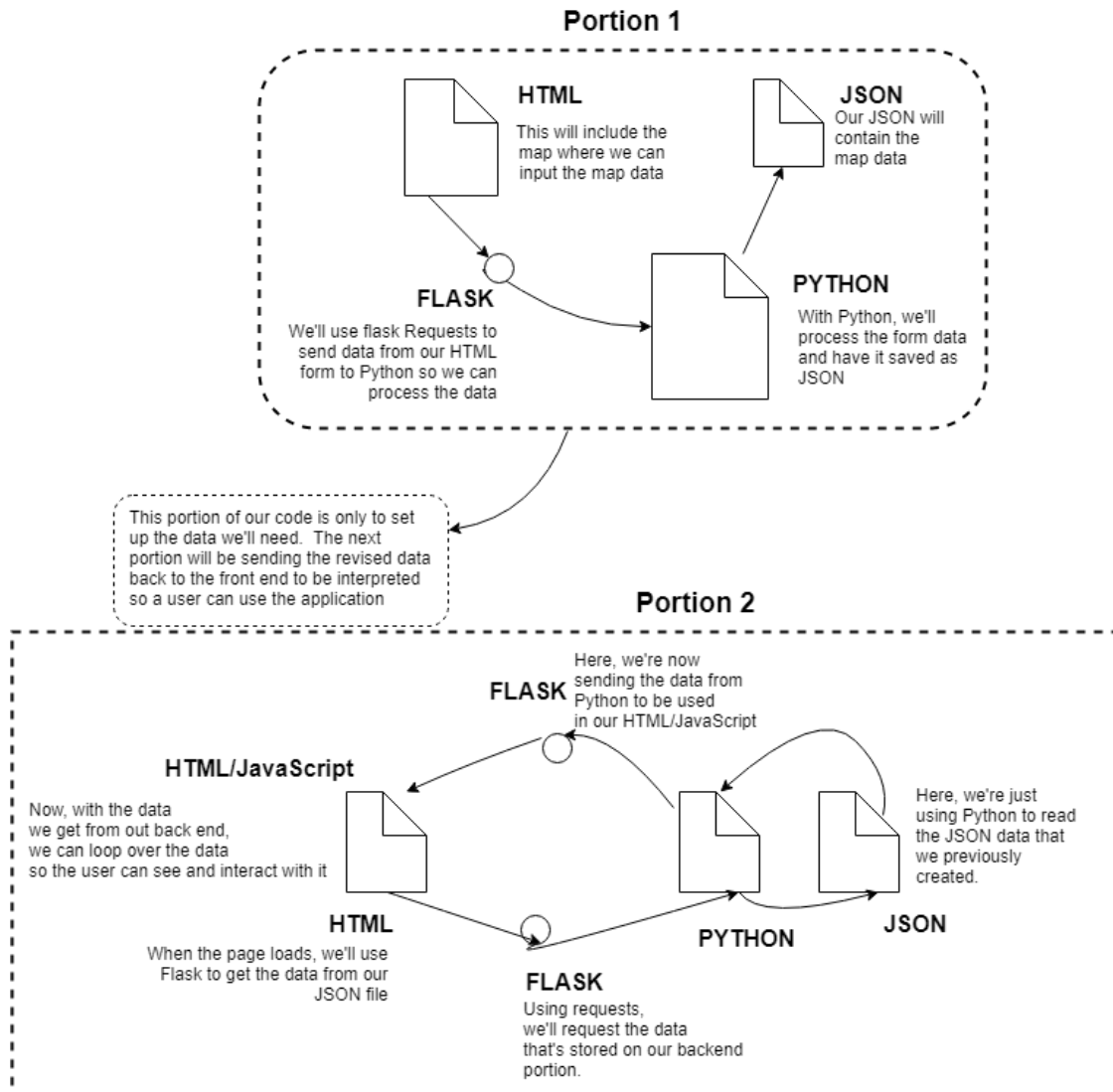


Figure 13. Flask Working

As far as the design on front end is concerned, it is done through HTML, CSS, and Vanilla JS. The website is made responsive to ensure ease of use in any dimension. The web app is hosted through *ngrok* which is an application to expose local servers on your system to the internet for easy hosting for developers. Ngrok was downloaded in the

Colab environment to create a dynamic link through which the web app could be accessed easily. Ngrok creates secure and temporary public URLs to expose local development servers to the internet. We used this tool to expose our system to the internet to be accessed easily through a public URL.

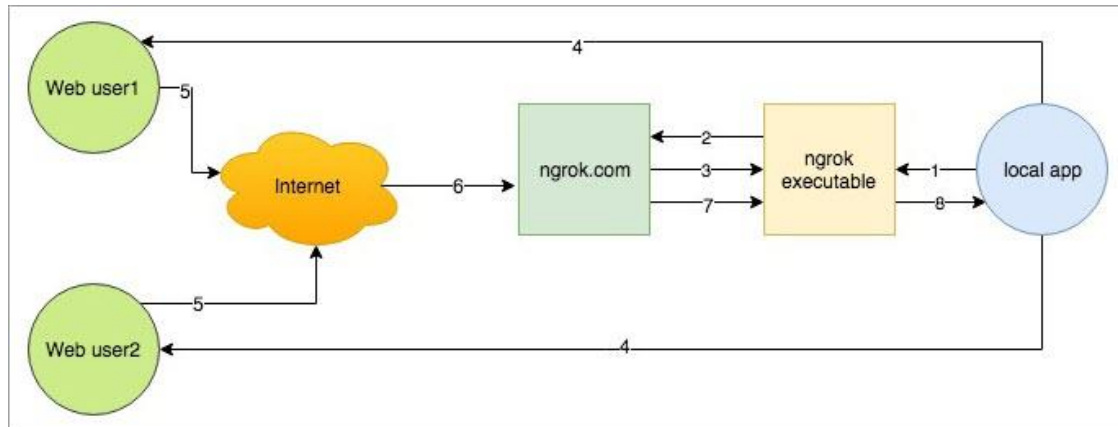


Figure 14. Work Flow of ngrok

2. API Model

The API model is a complete end-to-system deployed on the DubInc website. The user will upload any video file from their system onto the website. Then, when the user selects the API Model for dubbing, the video will be passed to the backend, along with user selection of API Model.

The model will extract audio from the video file and save it in an mp3 file called *extractedAudio.mp3*. This audio will then be transcribed, and the transcription will be saved within the system in form of a variable. The text will then be translated, and the translated text will also be stored within the system to be used by further modules. The system will synthesize speech from the Urdu text and store it in the local storage as an mp3 file called *dubAudio.mp3*. This final dubbed audio will then be superimposed with the original video to produce the final result in the form of a dubbed video called *resultVid.mp4*.

The folder structure for the intermittent outputs is shown in Fig 15 while the final output folder is shown by Fig 16.

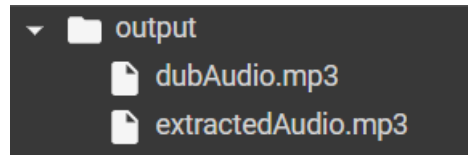


Figure 15. Output Folder

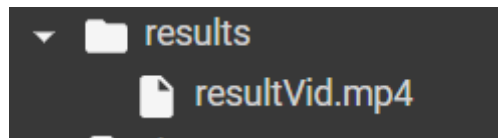


Figure 16. Result Folder

Architecture Design

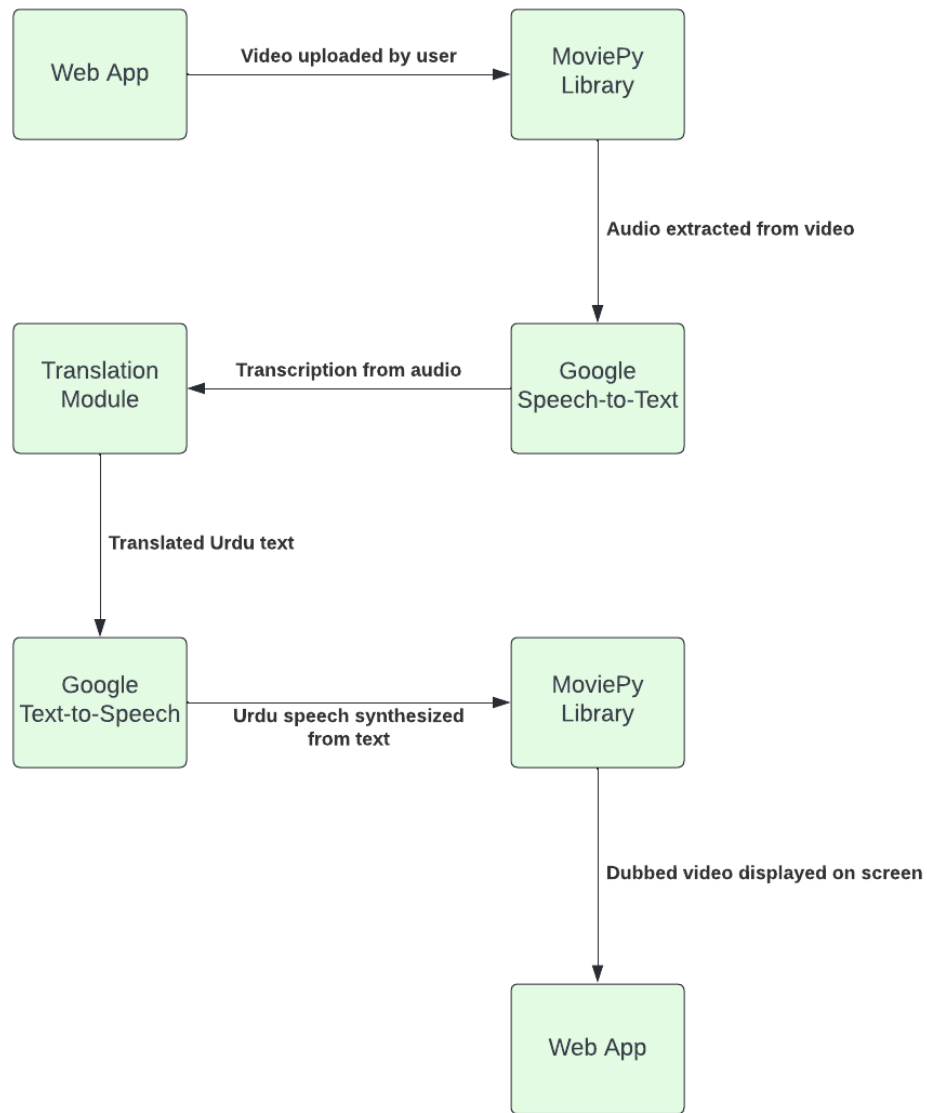


Figure 17. System Architecture

The figure depicts the overall architecture of our Automated Dubbing System that comprises of two main modules, the Web App, and the translation module. The translation module is then further divided into two subsystems, the API module and the

RNN module, only one of which is used at a time. The video to be dubbed will be provided to the system through the web app. The system will extract audio from the video. This audio will further be transcribed into text. The English text will then be translated by either the API module or the RNN/Custom Module depending on the user choice. This translated text will then be converted into audio which will in the end be superimposed on the original video by the system to generate a Dubbed Video file as an output.

b. DETAILED SYSTEM DESIGN

i. Classification

The major module of the system are the dubbing models utilized to dub any English video into Urdu.

ii. Definition

The various components of the system can be described as follows:

- **Web App**

The video file is uploaded and provided to the dubbing system at the backend. The Web App allows not only the upload of the video, but also selection of a dubbing model. For uploading the video, Flask form is used that allows users to upload the video file directly onto a specified directory, the local storage in this case. The system uses this video for processing and provides a dubbed video file in return which is also saved in the local storage and then displayed to the user through the Web App.

- **API Model**

The API model is used to dub the original English video into Urdu with the help of a python library called MoviePy and Google APIs, namely, the Speech-to-Text API, the Google Translate API, and the Text-to-Speech API.

iii. Responsibilities

- **Web App**

The web app provides an easy way for users to access and use the system for dubbing English videos into Urdu. It provides user with a simple way to upload the video to be dubbed. Afterward, user can also choose the type of dubbing model to dub the video i.e., either the API model or the Custom model.

Whichever model is chosen dubs the video, and then the web app displays the dubbed video for users to see. Users can also download the dubbed video through the website.

- **API Model**

The main responsibility of the API model is to dub whatever English video user has provided into Urdu. The system needs to use the video to extract its audio, transcribe it, translate, synthesize speech from it, and then submerge it with the original video for the final output.

iv. Constraints

- **Web App**

The website will only accept video files in the format *'mp4'*, *'mov'*, *'wmv'*, *'avi'*, or *'mkv'*. If user uploads a file outside of these extensions, the system will throw an error. The website also needs a video to process when the user tries to dub through the system. Therefore, if a file has not been uploaded, the choose file button on the website will show a warning to the user indicating the need for a file upload.

- **API Model**

The API model only works for dubbing of a video from English to Urdu. Therefore, if either the input video is not in English or it is desired to be dubbed in a language other than Urdu, the system will not be able to process the request.

v. Uses/Interactions

The API model and the Custom model are both hosted on the DubInc website. The website takes a video file as an input from the user. It then uses the python Flask framework to send a request to the backend to process the video. Depending on whether the user selected the API model option or the Custom model option, the processing request will be sent to one of these models. Each of these systems will extract audio from the video and then transcribe it. In case of API model, the Google Translate API will produce the Urdu translation. Whereas, for the Custom model, an RNN model will produce the translated Urdu text. Afterward, both these systems will synthesize speech from the text and combine it with original video to produce the final dubbed video.

vi. Resources

Prerequisites for API model include a service account on Google Cloud as well as a JSON access key.

vii. Interface/Exports

The libraries and technologies used for our system include following:

- **Web App**
 - i. Flask
 - ii. HTML
 - iii. CSS
 - iv. Ngrok
 - v. WTForms
 - vi. Flask Form

- vii. SubmitField
- viii. FileField
- ix. Werkzeug
- **API Model**
 - i. MoviePy
 - ii. Google-API-Core
 - iii. Google-Cloud-Speech
 - iv. JSON
 - v. OS
 - vi. SYS
 - vii. Google-Cloud-Storage
 - viii. Google-Cloud-Translate
 - ix. gTTs
 - x. VideoFileClip
 - xi. AudioFileClip
 - xii. CompositeAudioClip

IMPLEMENTATION AND TESTING

- **Technology Stack**

Web App	API Model	Custom Model
<ul style="list-style-type: none">• Flask• HTML• CSS• ngrok	<ul style="list-style-type: none">• Google Speech-toText• Google Translate• Google Text-to-Speech• MoviePy• gTTs	

- **Web App**

The web app has been integrated with Python backend through the use of Python framework called Flask. Flask forms along with HTTP POST and GET requests were used to handle the information passing between the frontend and the backend.

The web app is hosted using *ngrok* on a temporary and secure public URL for easy access.

RESULTS AND DISCUSSION

As we implemented our two models i-e API model and custom model, their output was measured across a confidence rate. Initially, the models that we used in our custom model had its own accuracy being measured and loss calculated. This was done with respect to the dataset that was varied in every training session. After training, our model on its own was tested on unseen and test data and their results compared with that of trained data. This was done in every architectural capacity. And as the model architecture was modified to acquire desired results, accuracy of the system was also observed to improve. This was also observed with the change in dataset and data preprocessing techniques. With appropriate data preprocessing techniques employed, a drastic change in the accuracy of the system was observed.

As described in the architectural details of the system, use of an RNN (recurrent neural network) allowed for the integration of an encoder-decoder model in our custom model where the dubbed voice was produced with translation from our own model. However, in our API model, the system employed a combination of various APIs to produce an output over the input video. These two varying approaches were measured against their performance by using a template video in English and dubbing it in Urdu.

In its final form, our model was able to produce 96% accuracy with minimal loss. It was also tested with various sentences of similar structure to observe if synonyms could be detected and translated accordingly. This venture was also found to be successful.

SIZE OF DATASET	ACCURACY
~2000 sentences	73%
~2000 sentences with preprocessing	96%
~15000 sentences	88%

CONCLUSION AND FUTURE WORK

As we explored the aforementioned problem statement, it soon became evident that lack of research in the machine translation domain for English-Urdu has adverse impacts on any potential solution that can be developed for this. Lack of dataset and abundance of it also originates from similar roots.

Therefore, if only future work can be focused on dataset creation and its analysis through existing or pre trained models, a significant improvement can be observed. Similarly, to work on models and to attain a better architecture in the future should also be focused on. Combining the two, a sufficiently accurate end to end system can be produced. Potential development on speech synthesis and Text to Speech (TTS) and more experimentation with it also promises phenomenal improvement on existing systems.

REFERENCES

- [1] "CSA Research," 7 July 2020. [Online]. Available: <https://csa-research.com/Blogs-Events/CSA-in-the-Media/Press-Releases/Consumers-Prefer-their-Own-Language#:~:text=65%25%20prefer%20content%20in%20their,information%20in%20their%20own%20language>. [Accessed 10 May 2022].
- [2] Y. Yang , B. Shillingford, Y. Assael, M. Wang, W. Liu, Y. Chen, Y. Zhang , E. Sezener, L. C. Cobo, M. Denil, Y. Aytar and N. d. Freitas, "Large-scale multilingual audio visual dubbing," 6 November 2020. [Online]. Available: <https://doi.org/10.48550/arXiv.2011.03530>. [Accessed 10 May 2022].
- [3] B. Wissmath, D. Weibel and R. Groner, "Dubbing or Subtitling?: Effects on Spatial Presence, Transportation, Flow, and Enjoyment," *Journal of Media Psychology Theories Methods and Applications*, vol. 21, no. 3, pp. 114-125, 2009.
- [4] C. M. Koolstra, A. L. Peeters and H. Spinhof, "The pros and cons of dubbing and subtitling," *European Journal of Communication*, vol. 17, no. 3, pp. 325-354, 2002.
- [5] C. Hu, Q. Tian, T. Li, W. Yuping, Y. Wang and H. Zhao, "Neural Dubber: Dubbing for Videos According to Scripts," *Advances in Neural Information Processing Systems*, vol. 34, 2021.
- [6] A. S. S. Z. H. Saad A. Bazaz, "Automated Dubbing and Facial Synchronization using Deep Learning".
- [7] S. Roxborough, "The Hollywood Reporter," 13 August 2019. [Online]. Available: <https://www.hollywoodreporter.com/tv/tv-news/netflix-s-global-reach-sparks-dubbing-revolution-public-demands-it-1229761/>. [Accessed 10 May 2022].
- [8] A. K. H. D. L. S. A. R. Shashi Pal Singh, "Machine translation using deep learning: An overview".
- [9] S. H. O. B. H. N. Thomas Deselaers, "A Deep Learning Approach to Machine Transliteration".

- [10] Y. W. X. C. Shuoheng Yang, "A Survey of Deep Learning Techniques for Neural Machine Translation".
- [11] M. T. J. T. Ł. K. J. U. O. B. & Z. Ž. Martin Popel, "Transforming machine translation: a deep learning system reaches news translation quality comparable to human professionals".
- [12] Y. L. & J. Zhang, "Deep Learning in Machine Translation".
- [13] A. S. R. T. Siddhant Srivastava, "Machine Translation : From Statistical to modern Deep-learning practices".
- [14] K. C. Y. B. Dzmitry Bahdanau, "Neural Machine Translation by Jointly Learning to Align and Translate".
- [15] D. X. J. S. Biao Zhang, "Neural Machine Translation with Deep Attention".
- [16] M. F. A. B. Alp Öktem, "Prosodic Phrase Alignment for Machine Dubbing".

