Media Engineering and Technology Faculty
German University in Cairo

Lecture Slides Annotation with Code-Switched Speech

Bachelor Thesis

Author: Yusuf Muhammad Eissa Ahmed Eissa Ammar

Supervisors: Dr. Nada Sharaf

Dr. Caroline Sabty

Submission Date: 12 June, 2022

Media Engineering and Technology Faculty
German University in Cairo

Lecture Slides Annotation with Code-Switched Speech

Bachelor Thesis

Author: Yusuf Muhammad Eissa Ahmed Eissa Ammar

Supervisors: Dr. Nada Sharaf

Dr. Caroline Sabty

Submission Date: 12 June, 2022

This is to certify that:

(i) the thesis comprises only my original work toward the Bachelor Degree

(ii) due acknowlegement has been made in the text to all other material used

Yusuf Muhammad Eissa Ahmed Eissa Ammar
12 June, 2022

Acknowledgments

I would like to thank and express my heartfelt gratitude to my professors and supervisors, Dr. Nada Sharaf Dr. Caroline Sabty, for their ongoing support, guidance, knowledge, and patience to be able to complete my research.

Abstract

In our day-to-day lives, we use automatic speech recognition in a variety of ways. We were able to simplify the learning process for students by utilizing this cutting-edge technology. This was done by making a system that is made up of 3 modules. The first module is the lecture viewing module that takes in the lecture slides in order to view them, record the code-switched (English Arabic) live speech of the lecturer, and record the timestamp of every slide switch. Then, the second module, speech to text module, takes in the audio recording of the lecturer to transcribe it and output text. The second module uses a post processing approach along with a speech recognition system in order to overcome the challenge of transcribing code-switched speech which can not be done by any of the most advanced speech recognition systems out there up to this date. Then, the third module, the slide matching module, takes in the text of the transcribed code-switched speech and the slide switching timestamps to output a new PDF file of the lecture slides where each slide is annotated with its corresponding explanation. The new annotated lecture slides can be viewed by the students to save them the trouble of taking notes themselves.

Contents

Chapter 1

Introduction

Education has drastically changed over the past years as advances in technology took place. Pandemic diseases like swine flu and COVID-19 also played a big role in changing the education system. There was big transition of replacing conventional studying material with digital content that can be accessed anywhere with a touch of a button. Lessons have become PowerPoint presentations, books have become PDF files, and lectures have become video recordings. Nowadays, students do not carry around books in their backpacks, instead they carry around their laptops, tablets, or smartphones.

Today, professors in universities use PDF files or PowerPoint presentations during lectures to explain the lesson. As these lecture slides contain only headlines that help the instructor explain the full lesson, students can't rely fully on these slides to study from later on. Students usually take notes of what the instructor says during class, so they can use it when studying for exams.

The aim of this project is to save students the trouble of taking notes themselves during lectures, and make them actually listen carefully to the lecturer instead of just taking notes blindly. This will be done using the help of speech recognition systems. This project also aims to overcome the challenge of transcribing code-switched speech as top-end speech recognition systems up to this date can only recognize speech with 1 language at a time.

This thesis includes 6 chapters. Topics related to this project like code-switching, speech recognition, machine learning, and deep learning are discussed in chapter 2, the background chapter. The system architecture and the modules that make up our system are discussed in chapter 3, the methodology chapter. The results of our project are discussed in Chapter 4, the results and discussion chapter. The conclusion of our project is discussed in chapter 5, the conclusion chapter. Further work needed to improve our system is discussed in chapter 6, the future work chapter.

Chapter 2

Background

In this chapter, background information about the technologies used in this project is presented.

2.1 Code Switching

When a speaker switches between two or more languages in the course of a single conversation or circumstance, this is known as code-switching or language alternation in linguistics. When multilinguals (those who speak more than one language) converse with one another, they may employ components of several languages.

Due to the frequency and variety of code-switching, it is possible to identify it as sentence alternation more frequently. It is possible for a sentence to begin in one language and end in another. Or, sentences from both languages may appear to follow one another at random. This behaviour may only be explained by a variety of linguistic or social elements, including the following: [1]

- When speakers are unable to explain themselves sufficiently in one language, they move to another to compensate.
- Using a minority language to demonstrate sympathy with a social group is fairly prevalent. The shift in language indicates to the listener that the speaker comes from a particular background.
- The attitude of the speaker toward the listener can be conveyed by switching languages like being friendly, annoyed, aloof, ironic, jocular, and so on. Monolinguals can transmit these effects to some extent by changing the formality of their speech, while bilinguals can do so by switching languages.

2.1.1 Types

- **Intersentential**: happens outside of the sentence or clause. [2] "Extrasentential" switching is another term for it.[3]
- **Intra-sentential:** within a sentence or a clause, switching occurs. [2][3]
- **Tag-switching:** is the process of changing the language of a tag phrase, a word, or both. In intra-sentential switches, this is a common occurrence. [2]
- **Intra-word:** switching takes place within a single word. [3]

2.2 Speech Recognition

Speech recognition, also known as automatic speech recognition (ASR), computer speech recognition, or speech-to-text, is a feature that allows a computer software to convert human speech into written text. While it is often confused with voice recognition, speech recognition is concerned with converting speech from a verbal to a text format, whereas voice recognition is just concerned with identifying the voice of an individual. [4]

2.2.1 Algorithms

The inconsistencies of human speech have made progress difficult. It is regarded as one of the most difficult disciplines of computer science, as it combines linguistics, mathematics, and statistics. The speech input, feature extraction, feature vectors, a decoder, and a word output are all parts of a speech recognizer. To identify the right output, the decoder employs acoustic models, a pronunciation dictionary, and language models. The accuracy rate of speech recognition technology, also known as the word error rate (WER), as well as its speed, are measured. Pronunciation, accent, pitch, loudness, and background noise are all characteristics that might affect word mistake rate. Speech recognition systems have long sought to achieve human parity, or an error rate comparable to that of two individuals speaking. To convert speech to text and increase transcription accuracy, a variety of algorithms and computer approaches are applied. The following are some of the most regularly utilised approaches, with brief descriptions:

- **Hidden Markov Models (HMM):** Hidden Markov Models are based upon that Markov chain model, which posits that the likelihood of a given state is determined by its present state, rather than its previous states. Hidden Markov models enable us to include concealed events, such as part-of-speech tags, into a probabilistic model. They are used as sequence models in speech recognition, giving labels to each item in the sequence—words, syllables, phrases, and so on. These labels form a mapping with the input, enabling it to choose the best label sequence. [5]

- **N-grams:** This is the most basic form of language model (LM), in which sentences or phrases are assigned probability. An N-gram is a collection of N words. "Order the pizza," for example, is a trigram or 3-gram, whereas "please order the pizza" is a 4-gram. To increase identification and accuracy, grammar and the likelihood of particular word sequences are employed. [4]

- **Neural networks:** Neural networks analyze training data by simulating the interconnection of the human brain through layers of nodes, which are mostly used for deep learning algorithms. Neural networks use supervised learning to learn this mapping function, then update it depending on the loss function via gradient descent. While neural networks are more precise and can handle more input, they have a lower performance efficiency than classic language models since they take longer to train. [5]

2.2.2 Applications

Speech technology is already being used in a broad range of sectors, allowing businesses and consumers to save time and even lives. Here are a few examples:

- **Automotive:** By providing voice-activated GPS navigation and search tools in vehicle radios, speech recognizers increase driving safety. [4]

- **Technology:** Virtual assistants are becoming more and more interwoven into our daily lives, especially on mobile devices. We use voice commands to control them via our smartphones, including Google Assistant or Siri (Apple), for tasks like voice search, or via our speakers, such as Alexa (Amazon) or Cortana (Microsoft), for music playback. They will only become further integrated into the items we use on a daily basis, fuelling the "Internet of Things" trend. [4]

- **Healthcare:** To capture and track patient diagnoses and treatment notes, doctors and nurses use dictation programs. [4]

- **Assisting The Visually And Hearing Impaired:** People with vision or hearing impairments can now type on computers and have text read to them aloud. [4]

- **Sales:** In sales, voice recognition has a handful of uses. It can assist a call centre in transcribing thousands of client and agent phone calls in order to detect typical call patterns and difficulties. Cognitive bots may also communicate with consumers through a website, answering common questions and completing simple requests without the need to queue for a contact centre employee. In both cases, voice recognition technology assist shorten the time it takes to resolve customer complaints. [4]

- **Security:** Security protocols are becoming increasingly vital as technology gets more embedded into our daily lives. Voice-based authentication offers an additional degree of security. [4]

2.3 Machine Learning

Machine learning is a branch of computer science that is distinct from standard computational methods. Algorithms are sets of clearly coded instructions that computers employ to compute or solve problems in classical computing. Machine learning algorithms, on the other hand, allow computers to learn from data inputs and employ statistical analysis to produce results that are within a certain range. As a result, machine learning makes it easier for computers to develop models from sample data and automate decision-making procedures based on data inputs. Machine learning has benefited everyone who uses technology today. Social media networks may utilise facial recognition technology to help users tag and share images of pals. The technology of optical character recognition (OCR) turns text pictures into moveable type. Machine learning-powered recommendation systems recommend what movies or TV series to watch next depending on user preferences. Consumers may soon be able to purchase self-driving cars that use machine learning to navigate. Machine learning is a discipline that is constantly evolving. As a result, there are a few things to think about while working with machine learning methodology or examine the effect of machine learning procedures. [6]

2.3.1 Methods

Tasks are often categorised into major groups in machine learning. These classifications are based on how training is processed and how the system is provided feedback on the learning. Unsupervised learning, which supplies the algorithm without any labelled data in order to enable it to identify pattern within its input data, is one of the most extensively used machine learning approaches. [6]

Supervised Learning

The computer is given sample inputs that are labelled with the expected outputs in supervised learning. The goal of this technique is for the algorithm to "learn" by comparing its real output to the "learned" outputs in order to detect faults and adjust the model accordingly. As a result, supervised learning predicts label values on unlabeled data using patterns. An algorithm could be given data containing photographs of sharks identified as fish and pictures of seas labelled as water, for example, using supervised learning. The supervised learning system should be able to recognise unlabeled shark photos as fish and unlabeled ocean photos as water after being trained with this data. Using previous data to forecast probable events in the future is a typical use of supervised learning. It might be used to predict future stock market changes or to filter spam emails using previous stock market data. Untagged photographs of dogs may be classified using labeled photos of dogs as input data in supervised learning. [6]

Unsupervised Learning

In unsupervised learning, data are not labelled, hence it is up to the learning algorithm to discover similarities among the unlabeled input data. Machine learning approaches that promote unsupervised learning are hugely helpful since unlabeled data is more available than

labelled data. The objective of unsupervised learning might be as simple as detecting underlying patterns within a dataset, or it could be feature learning, which enables the computational machine to autonomously discover the representations required for classifying raw data. For transactional data, unsupervised learning is often employed. You may have a vast dataset of consumers and their transactions, but as a human, you will be unable to deduce what related features may be derived from customer details and purchase kinds. With this information input into an unsupervised learning algorithm, it could be possible to deduce that women of a given age range who use odorless soaps are expected to be pregnant, and so a marketing campaign promoting pregnancy and infant items may be targeted to this demographic to enhance sales. Unsupervised learning approaches can look at complicated data that is more wide and seems unconnected without being given a "right" response in order to arrange it in potentially relevant ways without being given a "correct" answer. Unsupervised learning is frequently used in anomaly detection, such as for detecting credit card fraud purchases and recommendation systems that suggest what things to buy next. Untagged dog photographs can also be used as data input for an algorithm to discover likenesses and group dog photos together in unsupervised learning. [6]

2.3.2 Approaches

K-Nearest Neighbor

The k-nearest neighbour approach is a model for pattern recognition that may be utilised for classification and regression. The k in k-nearest neighbour is a positive integer that is usually small, and it is sometimes shortened as k-NN. The input will be the k closest training instances inside a space in either classification or regression. The result of this function is class membership. This will place a new object in the class with the most members among its k closest neighbours. The item is allocated to the class of the single nearest neighbour when k = 1. [6]

Decision Tree Learning

Decision trees are used to visually depict decisions and demonstrate or enlighten decision-making in general. Decision trees are utilised as a prediction model when dealing with machine learning and data mining. These models connect data observations to judgments about the target value of the data.The purpose of decision tree learning is to construct a model that can predict the value of a target based on input factors. The branches of the predictive model reflect the data qualities that are determined through observation, while the leaves represent the inferences about the target value of the data. When a tree is "learned," the original data is separated into subgroups based on an attribute value test, which is then performed recursively on each of the generated subsets. The recursion process will be finished when the subset at a node has the same value as its goal value. [6]

2.4 Deep Learning

2.4.1 How is deep learning different from machine learning?

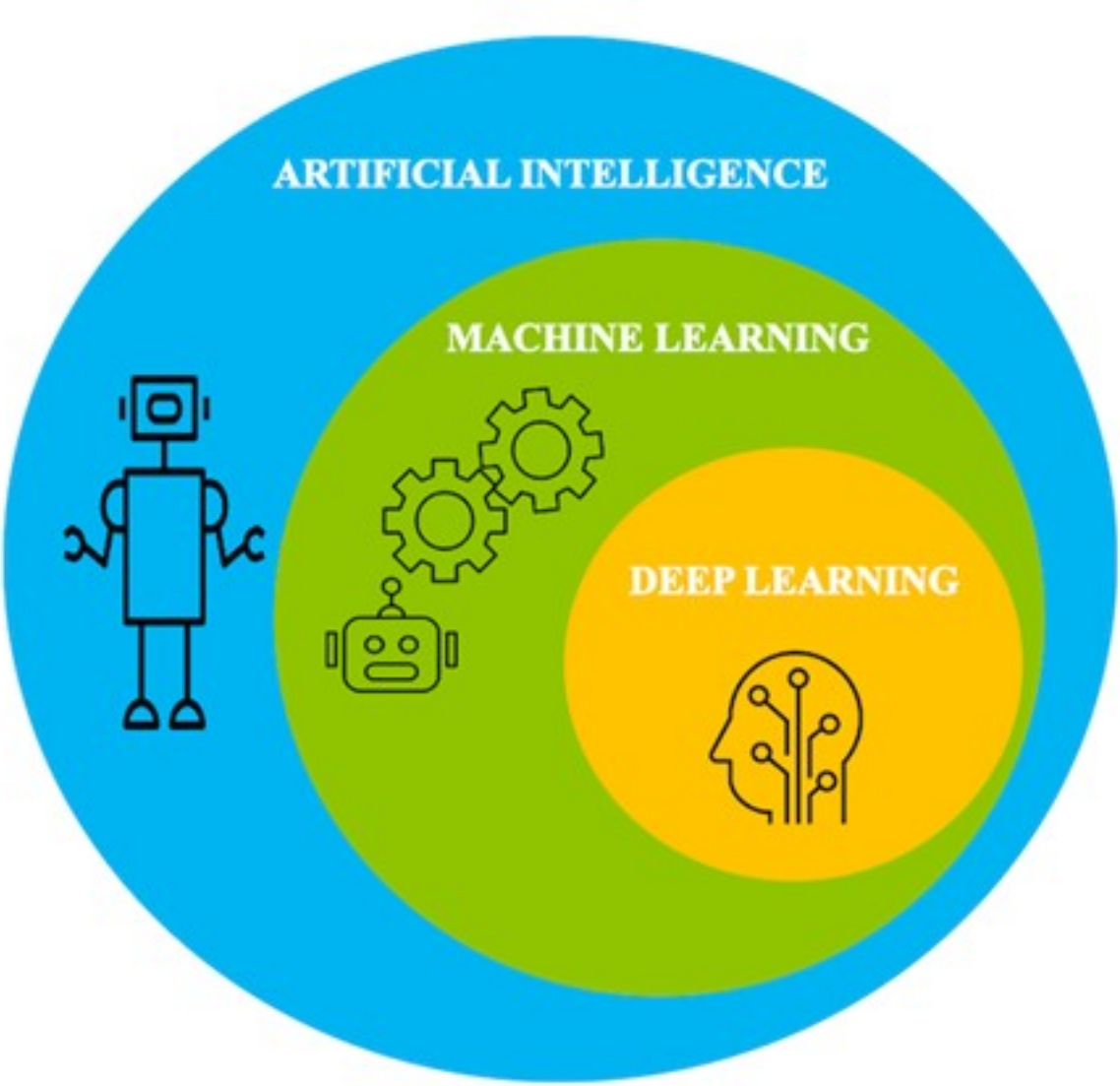

Figure 2.1: Machine Learning vs Deep Learning

[7]

Machine learning is a set of methods and tools that enable machines to recognise patterns in data and utilise this internal structure to reason about a problem. Machines attempt to comprehend these fundamental patterns in a variety of ways. But what is the relationship between machine learning and deep learning? Deep learning is being misunderstood as a competitive technique in the machine learning sector. Deep learning is a subcategory of machine learning. Deep learning has a wide range of applications, including health care, finance, and image

recognition. Deep learning algorithms may self-learn hidden patterns inside data to create predictions thanks to increased processing power and big data sets. Deep learning may be thought of as a subset of machine learning that is trained on enormous quantities of data and uses a large number of computer units to make predictions. [8]

2.4.2 Deep Learning and Human Brain Relation

The fundamental architecture of was inspired by the structure of the human brain in an effort to design systems that learn similarly to how people do. As a result, a number of essential terms in deep learning may be traced back to neurology. Deep learning architecture includes a computing unit called a perceptron that permits modelling of nonlinear functions, similar to how neurons compose the core components of the brain. The modest perceptron is where the magic of deep learning begins. The perceptron accepts a list of input signals and changes them into output signals, similar to how a "neuron" in the human brain delivers electrical pulses throughout the nervous system. The perceptron attempts to comprehend datasets by stacking many layers, each of which is responsible for comprehending a different aspect of the input. A layer is a group of computing units that learn to recognise a set of values that occur repeatedly. Every layer of perceptrons is in charge of deciphering a distinct pattern in the data. The design is dubbed neural networks because a group of these perceptrons mirrors the way neurons in the brain create a network. [8]

Neural Networks

A neural network has three layers in its most basic form: input layer, hidden layer, and output layer. A shallow neural network is one that has only one hidden layer. A shallow neural network with more than one hidden layer is called a deep neural network. Each hidden layer neuron is linked to a large number of others. Each arrow has a weight feature that governs how much the activation of one neuron impacts the others connected to it. Deep learning gets its name from these deep buried layers, and it gets its effectiveness from them. The number of hidden layers to use is determined by the nature of the problem and the size of the data collection. [8]

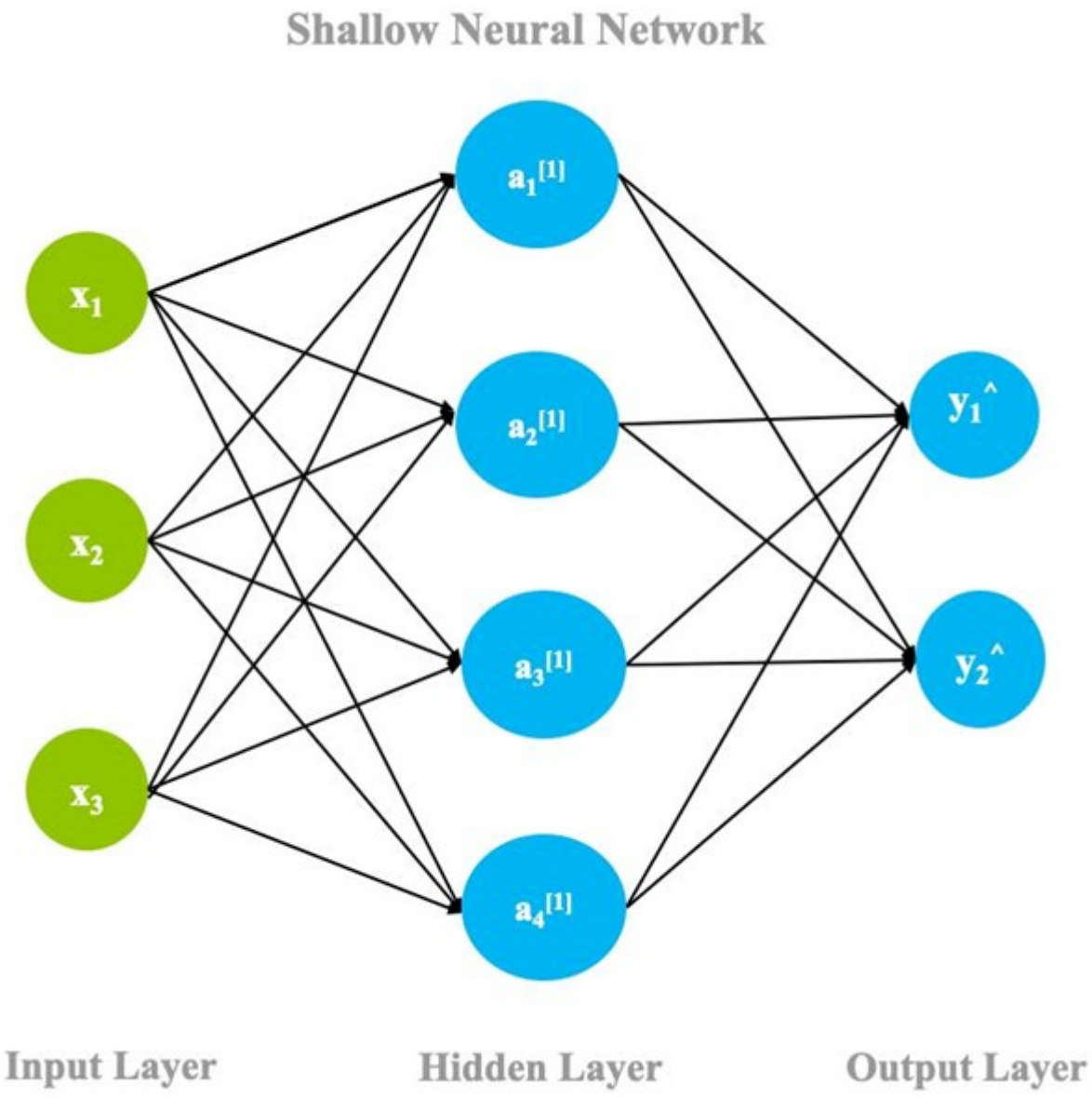

Figure 2.2: Shallow Neural Network

[7]

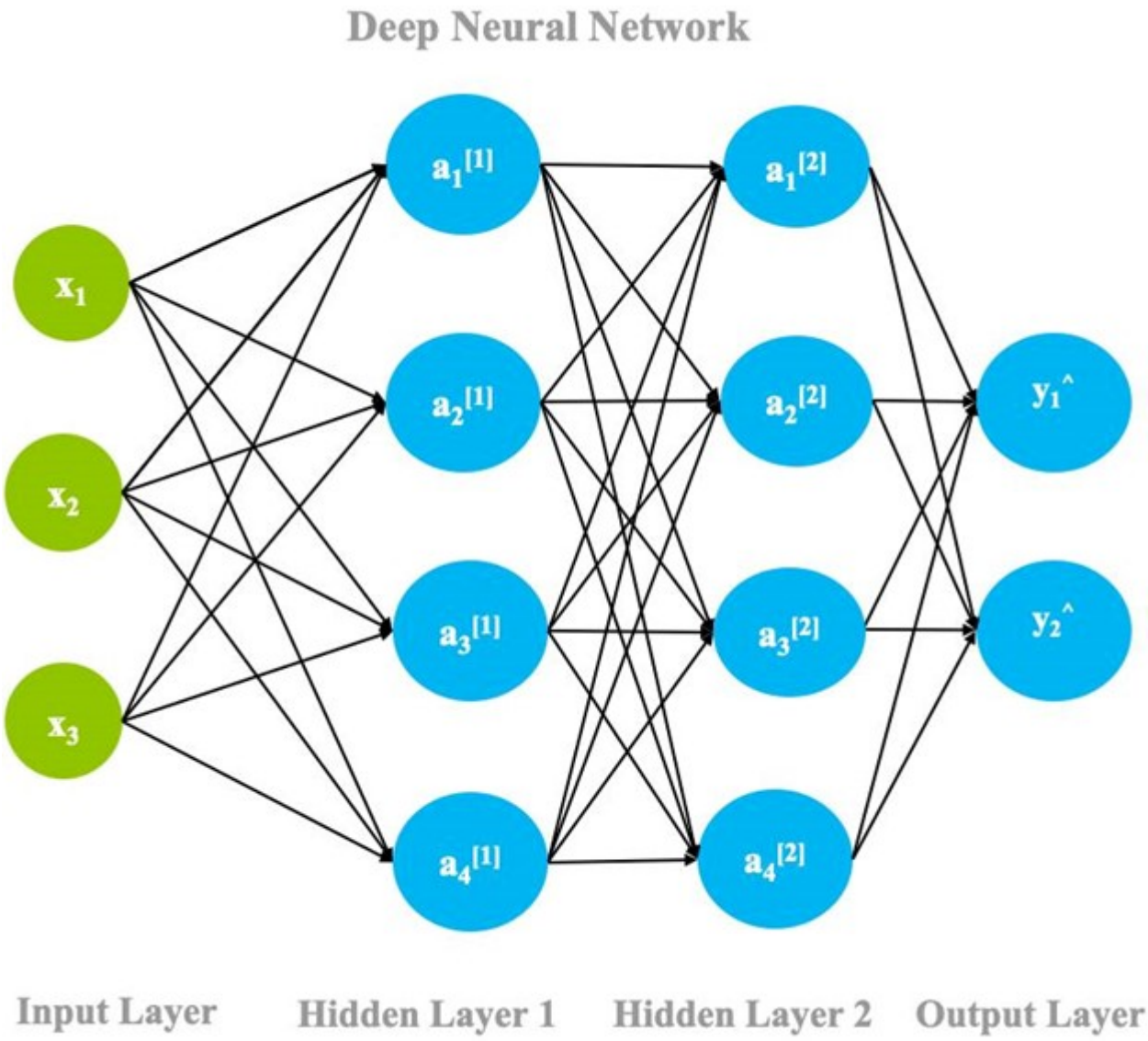

Figure 2.3: Deep Neural Network

[7]

Chapter 3

Methodology

In this chapter, the architecture of our system, the different modules of our system, and the different approaches used to implement every module is presented. The goal of our system is to transcribe the code switched speech of the lecturer and use it to annotate the lecture slides.

3.1 System Architecture

Our system is made up of 3 modules: the lecture viewing module, the speech-to-text module, and the slide matching module. The lecture viewing module is used to view the lecture PDF file and output an audio recording and the slide switching timestamps. The audio recording will passed on to the speech to text module, and the slide switching timestamps will be passed on to the slide matching module. Then, the speech to text module will take the audio recording as input and output text that will be passed on the slide matching module. Then, the slide matching module will take as input the text and the slide switching timestamps and output a new lecture PDF file with notes added to its slides.

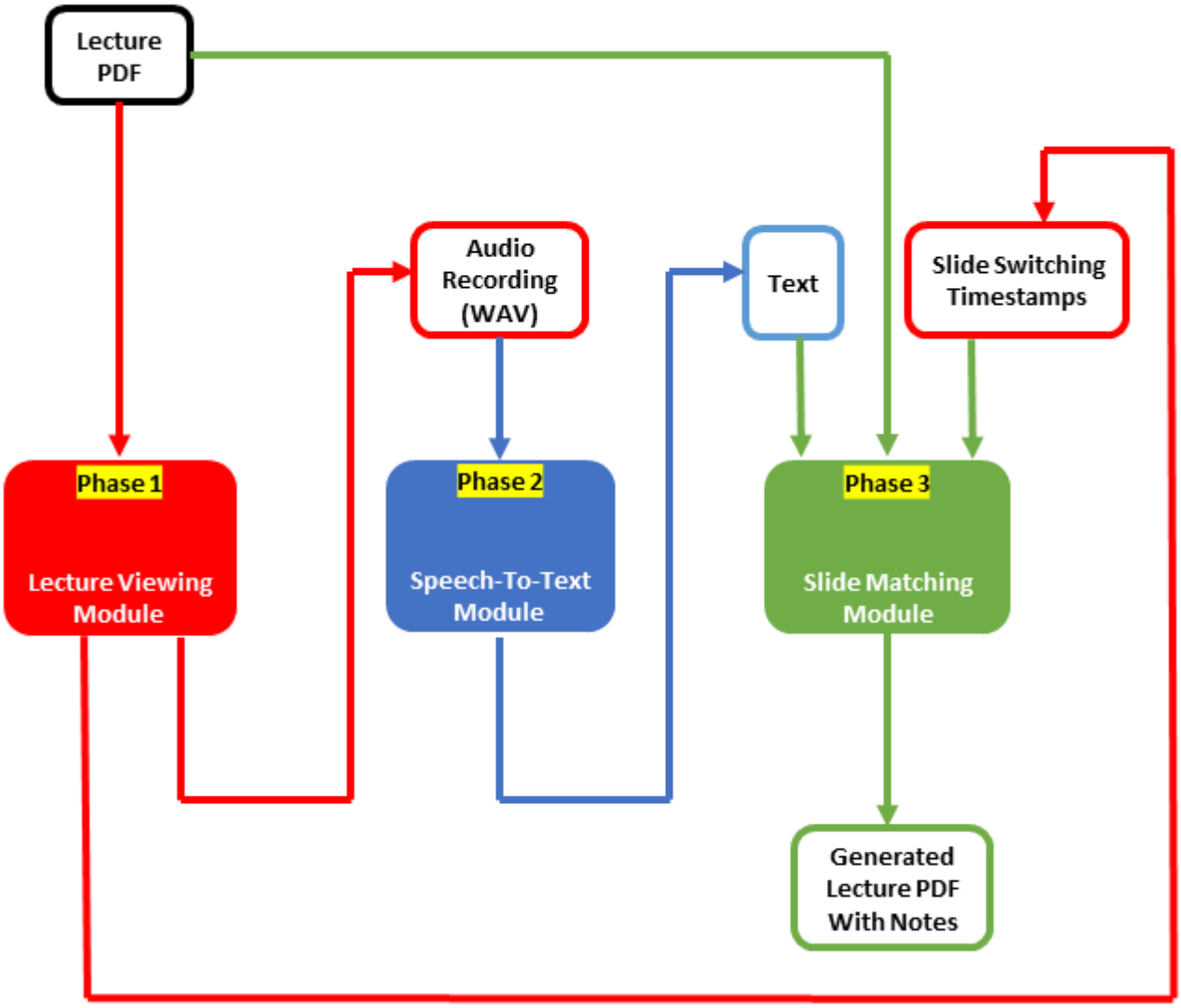

Figure 3.1: System Architecture

3.2 Phase 1 - Lecture Viewing Module

The lecture viewing module will be used by the lecturer to view and explain the lecture to his students. This module takes as input the lecture PDF file. The lecture viewing module is basically a website. The front end was implemented using HTML, and the back end was implemented using JavaScript and Python (flask library).

The lecture viewing module is made up of 3 components: PDF viewer, sound recorder, and stopwatch. The PDF viewer is used to view the lecture. The pdf viewer was implemented using a JavaScript library named pdf.js [9]. The sound recorder captures the live speech of the lecturer and saves it as an audio file (WAV). The sound recorder was implemented using a ready made JavaScript file called recorder.js [10], and was edited to output the audio file as a WAV

file. The stopwatch is a millisecond stopwatch that keeps track of the session time to be able to record the time of each slide switch.

The website (lecture viewing module) consists of 2 pages. In the first page, you upload the lecture pdf file you want to view. In the second page, the lecture you have uploaded is being viewed. The second page has a viewing box that shows 1 page at a time, page number of the slide currently viewed, number of total pages, and 3 buttons (record button, pause/resume button, and stop button). The record button starts a session where the sound recorder and stopwatch become active. The pause/resume button pauses or resumes the session. Changing the slide or page currently viewed is done through the arrow keys: right arrow key for previous slide and left arrow key for next slide. When the session is active and a slide switch occurs, the newly active slide number and time (in milliseconds) of the slide switch will be recorded. The stop button stops the session, and outputs the audio WAV file (recording of the live speech of the lecturer) and the slide switching timestamps (the newly active slide number and time (in milliseconds) of every slide switch).

The audio WAV file is passed then to phase 2 of our system, the speech to text module, and the slide switching timestamps are passed to phase 3 of our system, the slide matching module.

Figure 3.2: The lecture viewing module after taking the lecture (PDF) as input

3.3 Phase 2 - Speech-To-Text Module

This module will be used to convert the code-switched speech of the lecturer into text. This module take as input the audio file (WAV), passed on from phase 1 of our system: lecture viewing module.

One challenge we faced was that the explanation of the lecturer will be code-switched (English and Arabic), and there are no automatic speech recognition (ASR) system that can recognize code-switched speech. All available ASR sytems up to this date can only recognize 1 language at a time. To overcome this, post-processing approaches were used.

3.3.1 Automatic Speech Recognition (ASR) System Selection

In order to select which ASR system to use, multiple speech recognition libraries and speech recognition API(s) were tested by 20 sample audio files: 10 with English-only speech and 10 with Arabic-only speech.

The difference between speech recognition libraries and speech recognition API(s) is that the libraries are free to use, while the API(s) charge you to use them.

The speech recognition libraries tested were VOSK and CMUSphinx, and the speech recognition API(s) tested were Google Cloud Speech To Text and Microsoft-Azure Speech To Text. Speech Recognition API(s) were better than the open source libraries in terms of WER (word-error rate) especially when it comes to recognizing Egyptian-Arabic speech. Microsoft-Azure Speech To Text and Google Cloud Speech To Text API(s) had similar accuracies, but Microsoft Azure Speech To Text API was chosen because it was easier to set up.

Microsoft-Azure Speech To Text API Configuration/Inputs

- Subscription Key and Region: required to be able to use the API
- Audio File (WAV) Path: to specify which audio file to be speech recognized
- Language Code: to specify the language + region/locale to be recognized from the audio file
- Enabling Word-Level Confidence: percentage of how sure the API is of each word, used in filtration of the text in the post-processing approaches
- Enabling Word-Level Timestamps: timestamp/ offset of each word, used in filtration of the output text in the post-processing approaches
- Enabling Profanity Filter: to remove profane words recognized due to misinterpretation of speech
- Enabling Detailed Output Format: to get output in JSON format with the specified properties (World-Level Confidence and Timestamps) to be able to do post processing
- Using Continuous Speech Recognition Method: to guarantee recognition of long audio files with no problems and not ending speech speech recognition if silence is detected. Unlike the recognize once method which was tested initially and had high WER (word error rate) due to words being skipped due to manual splitting of the audio file to overcome the stopping of speech recognition if silence is detected.

3.3.2 Approach 1: Continuous Speech Recognition With Continuous Language Identification

- **Step 1: Recognizing Audio File Using Continuous Language Identification**: The whole audio file will be recognized by enabling the Continuous Language Identification feature provided by the Microsoft Speech To Text API. This feature makes the API identify which language is being spoken then transcribes the audio, and this process keeps on repeating until the whole audio file gets transcribed. To enable this feature, the candidate languages (the possible languages of the speech included in the audio file being recognized) were specified. In our case, the candidate languages were English and Arabic.
- **Step 2: Output**: the text along with the timestamp/offset and confidence level for each word will be passed on to phase 3 of our system, the slide matching module.

3.3.3 Approach 2: Continuous Speech Recognition With Overlap Filtration

- **Step 1: Recognizing Each Language Separately**: The whole audio file is being recognized by the English and Arabic speech recognizers. Also, words with confidence level less than **5 percent** are eliminated.
- **Step 2: Merging Outputs**: Outputs of the English and Arabic speech recognizers are merged, and words are ordered by their timestamps.
- **Step 3: Overlap Filtration**: Sometimes 1 word gets recognized by both the English and Arabic speech recognizers due to similar pronunciation. To eliminate the wrong word, overlap filtration is done. Overlap filtration is done by comparing the timestamp and duration of every 2 consecutive words. If the overlap percentage between any 2 words is more than 40 percent (this number was obtained by trial and error), the word with the lower confidence would be eliminated if and only if the confidence difference between the 2 words is greater than 5 percent. Otherwise, none of the words will be eliminated due to insignificant confidence deference which means that there is a 50 percent chance that any of the words is the correct one. Note that it is better to add an incorrect word than to delete a correct word to increase the understandablility of the lecture notes.

$$Overlap = FirstWordOffset + FirstWordDuration - SecondWordOffset$$

$$OverlapPercentage = Overlap/SecondWordDuration$$

Figure 3.3: Overlap Calculation

- **Step 4: Output**: the text along with the timestamp/offset and confidence level for each word will be passed on to phase 3 of our system, the slide matching module.

3.4 Phase 3 - Slide Matching Module

The slide matching module will be used to segment the speech of the lecturer and match every segment to the correct slide. This module takes as input the lecture PDF file, text with the timestamp and confidence level of each word (passed on from phase 2 of our system, speech to text module), and the slide switching timestamps (passed on from phase 1 of our system, lecture viewing module).

3.4.1 Approach 1: Time-Alignment And Sentence Similarity Using Deep Learning

- **Step 1: Time Alignment**

  First, text will be split here into chunks using the slide switching timestamps (includes the newly active slide number) and the timestamps of each word. Second, chunks will be matched to each slide correspondingly. Time Alignment matches text to the slide that was viewed at the time of the text was said. After this step, every slide will have the text said when the slide was viewed, regardless of the contents of the slide.

  Slide matching using time alignment is needed because in previous approaches we split text if we detect a silence of 5 seconds but this was wrong because some speakers would pause for 2 seconds and move to the next slide while another speaker would wait for 3 or 4 seconds, and even the same speaker would wait for 5 seconds to move to the next slide in the beginning of the session and later on in the same session he/she would wait only for 3 seconds to move to another slide. Not knowing the start and end of each chunk or when the speaker moved to the next slide or when the speaker starting talking about a new topic will affected the slide matching significantly. For example, if you have 2 slides each talking about a relatively different topic and text supposed to be split into 2 chunks (one for each slide). Due to not knowing the start and end of each chunk the text gets split into 3 chunks: first chunk addresses one topic and gets matched correctly to the first slide, second chunk addresses 2 topics and gets matched to the first slide (while having text related to the second slide), and third chunk addresses one topic incompletely and gets matched incorrectly to the first slide due to having incomplete text

- **Step 2: Sentence Similarity Using Deep Learning**

  First, the notes matched to every slide will get split on silences that last for more than 5 seconds. By trial and error, 5 seconds silence is long enough to ensure the speaker started talking about another topic. Second, text will be extracted from the Lecture (PDF) using a Python library called fitz, where an array of strings is outputted (1 string for every slide). Then, a deep learning model will be used to match the smaller chunks to the slides. All the smaller chunks of each slide will be compared to the text of all the slides, using a sentence similarity deep learning model and cosine similarity in an attempt to find a better match for each smaller chunk.

The deep learning model used is the Pyjay/sentence-transformers-multilingual-snli-v2-500k [11]. This model is a bert-base model and loaded using a python library called Sentence-Transformers [12]. The model converts sentences and paragraphs into a dense vector space with 768 dimensions, which can be utilized for applications like clustering and semantic search. This model is a multilingual model that is used to find sentence similarity between code-switched text and English text. The model takes as input a reference sentence and sentences to be compared to the reference sentence. The model encodes the sentences and creates embeddings. Then, cosine similarity is used to get a similarity score for the sentences compared to the reference sentence by using the embeddings created which was done using a Python library called cosine-similarity.

This model was chosen by making a test set of 100 code-switched sentences, 100 English sentences, and 10 slides to test the accuracy of 9 multilingual models and 2 English-Only models in matching the sentences to the text of the slides. For matching the code switched sentences to the slides, the highest accuracy of 87 percent was obtained by the Pyjay/sentence-transformers-multilingual-snli-v2-500k (multilingual) model. For matching the English-only sentences to the slides, the highest accuracy of 85 percent was obtained by the all-mpnet-base-v2 (English-Only) model , and the second-highest accuracy of 83 percent was obtained by the Pyjay/sentence-transformers-multilingual-snli-v2-500k (multilingual) model. So, the better option was to use the Pyjay/sentence-transformers-multilingual-snli-v2-500k model for all sentences because it has the highest accuracy when it comes to matching code-switched sentences and a 2 percent difference in the accuracy of matching English sentences compared to the all-mpnet-base-v2 model is not significant. Also, if one model is used for code-switched sentences and another model is used for English sentences, this requires loading a model for every sentence separately which increases processing time by 10 times.

Slide matching done by getting sentence similarity using deep learning is needed because it is possible that the lecturer explains something not related to the viewed slide at that time. For example, a student asks the lecturer about something not related to the viewed slide at that time, and the lecturer answers his question without going back to the slide related to the question of the student. So we need to make sure the notes matched to the slides are the best match.

- **Step 3: Generating New PDF**

  New lecture PDF file will be generated with the matched notes appearing in the notes box of their corresponding slides.

3.5 Equipment

The equipment, specifications, and formats utilised in this thesis are described in this section.

- Laptop:

  - Acer Swift SFX14-41G
  - Processor: AMD Ryzen 7 5800U with Radeon Graphics
  - GPU: Nividia GeForce RTX 3050 TI
  - RAM: 16.00 GB
  - System Type: x64-based PC
  - OS: Windows 11
- Python 3.9.7
- JupyterLab 3.2.1: for writing Python code
- Microsoft Visual Studio 2019: for writing HTML and JavaScript code.
- Microphone Array (AMD Audio Device) (Built-In Laptop Microphone)
- Microsoft Azure: Speech To Text API
- Microsoft Azure: Translation API
- Audacity Software: to convert audio files from m4a to wav.

Chapter 4

Results and Discussion

In this chapter, the results and procedure of our testing phase along with the metrics used to measure the accuracy of our system and modules is demonstrated.

4.1 Word Error Rate (WER) Metric

Word Error Rate (WER) is the standard metric used to evaluate the accuracy of the automatic speech recognition (ASR) systems in terms of speech to text conversion. Top-end speech recognition systems like Microsoft and Google use the WER metric to evaluate their speech to text conversion. Microsoft has a WER of 5.1%, and Google has a WER of 4.9%. While, human transcribers have an average WER of 4%.

The lower the WER, the higher the accuracy of the speech recognition system. WER tends to get higher if the speaker has a bad accent, domain-specific terms are included in the speech, or bad audio quality due to using a bad microphone or surrounding noise.

WER is calculated by getting the number of words mistakenly added, subtituted, and deleted over the number of total words actually spoken.

4.2 Speech To Text Module Accuracy Results

The speech to text module used the Microsoft Azure - Speech To Text API combined with different post-processing approaches to be able to convert code-switched speech into text. The accuracy of every approach was evaluated using the Word Error Rate (WER) metric, and the results are presented below. Every approach was tested using our test set.

4.2.1 Test Set

A test set was constructed to test the speech to text module of our system. To make the testing as natural as possible, there were no constraints imposed on the audio files in terms of pace and accent of the speaker. The test set is made up of 11 audio files: 8 short code switched audio files, 1 long code switched audio file, and 2 English-only audio files. All audio files were WAV files.

- The 8 short code switched audio files were 3-6 secs long. They were named as "CS-Short1.wav", "CS-Short2.wav", "CS-Short3.wav", "CS-Short4.wav", "CS-Short5.wav", "CS-Short6.wav", "CS-Short7.wav", and "CS-Short8.wav" respectively. Each audio file was made up of 1 Intra-sentential code switched sentence, where the dominant language was English and 1 word or phrase of each sentence was in Arabic. These audio files were recorded with my voice.
- The long code switched audio file was 13 minutes long. It was named as "CS-Long.wav". The audio file was made up of 100 Intra-sentential code switched sentences, where the dominant language was English and 1 word or phrase of each sentence was in Arabic. This audio file was the the audio recording of the live lecture explanation (done by me using our system) during the testing of the slide matching module.
- The 2 English-Only audio files were extracted from an Embedded Systems course lecture, taught by Dr. Hassan Soubra, that was video recorded and uploaded on the Video on Demand service of the German University in Cairo. The first audio file was 1 minute 15 seconds long and named as "English-Short.wav". The second audio file was 12 minutes 30 seconds long and named as "English-Long.wav". These 2 audio files had English-only speech. The lecturer is French, so he is not a native English speaker.

4.2.2 Approach 1: Continuous Speech Recognition With Continuous Language Identification

All the audio files in our test set were tested. The WER, duration, total words, number of words that has been substituted, deleted, or added of each audio file is presented below. For more information about the audio files like the speaker, what type of code switching does it have, which language is spoken, or its origin please refer to the test set section above.

Short Code Switched Audio Files

- The first audio file, "CS-Short1.wav", was 6 seconds long. It had 20 total words. 5 words were substituted, deleted, or added which equates to a WER of 25%.
- The second audio file, "CS-Short2.wav", was 5 seconds long. It had 10 total words. 0 words were substituted, deleted, or added which equates to a WER of 0%.

- The third audio file, “CS-Short3.wav”, was 4 seconds long. It had 9 total words. 4 words were substituted, deleted, or added which equates to a WER of 44.4%.
- The fourth audio file, “CS-Short4.wav”, was 2 seconds long. It had 6 total words. 2 words were substituted, deleted, or added which equates to a WER of 33.3%.
- The fifth audio file, “CS-Short5.wav”, was 3 seconds long. It had 8 total words. 2 words were substituted, deleted, or added which equates to a WER of 25%.
- The sixth audio file, “CS-Short6.wav”, was 4 seconds long. It had 8 total words. 1 word was substituted, deleted, or added which equates to a WER of 12.5%.
- The seventh audio file, “CS-Short7.wav”, was 5 seconds long. It had 10 total words. 2 words were substituted, deleted, or added which equates to a WER of 20%.
- The eighth audio file, “CS-Short8.wav”, was 4 seconds long. It had 11 total words. 1 word was substituted, deleted, or added which equates to a WER of 9%.

The short code switched audio files have an average WER of 21%.

Audio File	WER
CS-Short1.wav	25%
CS-Short2.wav	0%
CS-Short3.wav	44.4%
CS-Short4.wav	33.3%
CS-Short5.wav	25%
CS-Short6.wav	12.5%
CS-Short7.wav	20%
CS-Short8.wav	9%
Average	**21%**

Figure 4.1: WER of the Short Code Switched Audio Files

Long Code Switched Audio File

The long code switched audio file, “CS-Long.wav”, was 12 minutes long. It had 1167 total words. 287 words were substituted, deleted, or added which equates to a WER of 24.6%.

English-Only Audio Files

- The first audio file, “English-Short.wav”, was 1 minute 15 seconds long. It had 155 total words. 4 words were substituted, deleted, or added which equates to a WER of 2.6%.

- The second audio file, “English-Long.wav”, was 12 minute 30 seconds long. It had 1591 total words. 28 words were substituted, deleted, or added which equates to a WER of 1.76%.

The English-Only audio files have an average WER of 2.18%.

Audio File	WER
English-Short.wav	2.6%
English-Long.wav	1.76%
Average	**2.18%**

Figure 4.2: WER of the English-Only Audio Files

4.2.3 Approach 2: Continuous Speech Recognition With Overlap Filtration

All the audio files in our test set were tested. The WER, duration, total words, number of words that has been substituted, deleted, or added of each audio file is presented below. For more information about the audio files like the speaker, what type of code switching does it have, which language is spoken, or its origin please refer to the test set section above.

Short Code Switched Audio Files

- The first audio file, “CS-Short1.wav”, was 6 seconds long. It had 20 total words. 0 words were substituted, deleted, or added which equates to a WER of 0%.
- The second audio file, “CS-Short2.wav”, was 5 seconds long. It had 10 total words. 2 words were substituted, deleted, or added which equates to a WER of 20%.
- The third audio file, “CS-Short3.wav”, was 4 seconds long. It had 9 total words. 0 words were substituted, deleted, or added which equates to a WER of 0%.
- The fourth audio file, “CS-Short4.wav”, was 2 seconds long. It had 6 total words. 1 word was substituted, deleted, or added which equates to a WER of 16.6%.
- The fifth audio file, “CS-Short5.wav”, was 3 seconds long. It had 8 total words. 1 word was substituted, deleted, or added which equates to a WER of 12.5%.
- The sixth audio file, “CS-Short6.wav”, was 4 seconds long. It had 8 total words. 1 word was substituted, deleted, or added which equates to a WER of 12.5%.
- The seventh audio file, “CS-Short7.wav”, was 5 seconds long. It had 10 total words. 0 words were substituted, deleted, or added which equates to a WER of 0%.

- The eighth audio file, “CS-Short8.wav”, was 4 seconds long. It had 11 total words. 1 word was substituted, deleted, or added which equates to a WER of 9%.

The short code switched audio files have an average WER of 8.8%.

Audio File	WER
CS-Short1.wav	0%
CS-Short2.wav	20%
CS-Short3.wav	0%
CS-Short4.wav	16.6%
CS-Short5.wav	12.5%
CS-Short6.wav	12.5%
CS-Short7.wav	0%
CS-Short8.wav	9%
Average	**8.8%**

Figure 4.3: WER of the Short Code Switched Audio Files

Long Code Switched Audio File

The long code switched audio file, “CS-Long.wav”, was 12 minutes long. It had 1167 total words. 155 words were substituted, deleted, or added which equates to a WER of 13.3%.

English-Only Audio Files

- The first audio file, “English-Short.wav”, was 1 minute 15 seconds long. It had 155 total words. 9 words were substituted, deleted, or added which equates to a WER of 5.8%.
- The second audio file, “English-Long.wav”, was 12 minute 30 seconds long. It had 1591 total words. 218 words were substituted, deleted, or added which equates to a WER of 13.7%.

The English-Only audio files have an average WER of 9.75%.

Audio File	WER
English-Short.wav	5.8%
English-Long.wav	13.7%
Average	**9.75%**

Figure 4.4: WER of the English-Only Audio Files

4.2.4 Results Evaluation

Audio File	Approach 1 (WER)	Approach 2 (WER)
Short Code Switched (avg)	21%	8.8%
Long Code Switched	24.6%	13.3%
English-Only (avg)	2.18%	9.75%

Figure 4.5: WER Approaches Comparison

As shown in the figure above, approach 2 performs better for all code switched audio files. However, for the English-only audio files approach 1 performs better.

We chose approach 2 to be used for the speech to text module in the next testing phase where we test the system as a whole because our main objective of this research in terms of speech recognition is to recognize code switched speech, and there is a big difference in WER between the 2 approaches when recognizing code switched speech in favor of approach 2.

4.3 Slide Matching Module Accuracy Results

In this testing phase, we are going to test our slide matching module while testing our system as a whole where all the modules (the lecture viewing module, the speech to text module, and the slide matching module) work together. The speech to text module will be using approach 2 (continuous speech recognition with overlap filtration) as we established that this approach resulted in the best results in terms of turning code switched speech into text. The slide matching module will be using the only proposed approach which is the time alignment and deep learning sentence similarity approach. The accuracy of our system will be expressed in terms of the accuracy of turning speech to text (evaluated using the WER metric) and the accuracy of matching text to slides (evaluated by getting the number of correctly match slides over the total slides).

4.3.1 Testing Procedure

To test our system as a whole we made a mock lecture by making a PowerPoint presentation of 10 slides. The PowerPoint presentation contained 100 English-Only sentences (1167 words) distributed over the 10 slides. Every slide addressed a relatively different topic and contained roughly 10 sentences. In a real-life lecture environment, the lecturer does not use the same words on the slides to explain the lecture and also uses code-switching (English and Arabic) to better explain the lecture. In order to mock that environment, a script was made by paraphrasing and code switching the 100 sentences: the text of the slides of our created lecture. The code switching done to the sentences was Intra-sentential, and it was done by translating 1 word or phrase of each sentence to Arabic.

Then, we used the interface of our system or the website to view the mock lecture PDF file, go through its slides, and explain it with the script created (the paraphrased code-switched script) using my voice.

4.3.2 Testing Results

There were 2 tests done to measure how good our system will perform under different conditions. The first test was the easier in terms of slide matching. The first test was done by going through the slides and saying the parts of the script that was related to the currently viewed slide. The second test was done by going through the slides and saying parts of the script that was not related to the currently viewed slide to test how good the module will perform when the lecturer says something not related to the currently viewed slide.

Both tests resulted in 13.3% WER as the accuracy of turning speech to text where the script had 1167 total words, and 155 words were substituted, deleted, or added.

Also, both tests resulted in 90% accuracy of matching text to slides where 9 slides out of the 10 slides were correctly matched.

4.4 Discussion

The speech to text module testing results were encouraging for both code switched sentences and English-only sentences. The second approach (continuous speech recognition with overlap filtration) outperformed the first approach (continuous speech recognition with continuous language identification). The first approach had higher WER than then second approach when transcribing code switched audio files, because the first approach uses language identification that works on relatively long silences that do not match the way of how people code switch. The downside of the second approach is that it tends to add words to the text if there was a word in the Arabic language that is similarly pronounced like a word in the English language or vice versa, so both words from both languages get added to the text while one of them is not the actual word that was spoken. That is why the second approach got outperformed by the first approach when English-only audio files with a non-native English speaker were tested. Due to the bad accent of the speaker, words were mistakenly recognized from the Arabic language and added to the text resulting in higher WER.

The best configuration of our system was to use the continuous speech recognition with overlap filtration approach (second approach) for the speech to text module along with the time alignment and sentence similarity deep learning approach for the slide matching module. This configuration resulted in 13.3% WER for turning code switched speech to text and a 90% accuracy for matching text to slides. These results encourage us to actually use this system in a real-lecture setting to make the life of students easier by saving them the trouble of writing notes during the lecture and actually listen to the lecturer.

Chapter 5

Conclusion

Education has changed dramatically in recent years as a result of technological advances. There has been a significant shift toward replacing traditional study materials with digital content that can be accessed with a touch of a button anywhere. During lectures, professors today use PDF files or PowerPoint presentations, and students can't fully rely on these lecture slides to study from later on because they only contain headlines.

Students usually take notes during class to use when studying for exams, so we aimed in this project to make the life of students easier and save them the hassle of taking notes during lectures, which was done with the help of a speech recognition system. We had to overcome the challenge of transcribing code-switched speech, as top-tier speech recognition systems can't do that as of now.

Our developed system is made up of three modules. The first module is the lecture viewing module, which receives the lecture slides in order to view them, records the lecturer's code-switched (English Arabic) live speech, and records the timestamp of each slide switch. The second module, speech to text, takes in the lecturer's audio recording and transcribes it into text. The second module employs a post-processing approach in conjunction with a speech recognition system to overcome the challenge of transcribing code-switched speech. The third module, the slide matching module, uses the text of the transcribed code-switched speech and the slide switching timestamps to generate a new PDF file of the lecture slides, annotated with the corresponding explanation for each slide.

Results shows that our system achieves a WER of 13.3% in transcribing code-switched audio files and an accuracy of 90% in matching text to slides. These results shows that with some enhancements to our system, our system can be actually used on the production level.

Chapter 6

Future Work

The results of our system were encouraging, but it needs some work to able to actually use it on the production level. Our system faced some sort of difficulty when transcribing English-only audio files that contained words that are similarly pronounced in the Arabic language. Words of the Arabic language were mistakenly recognized and added to the transcription, and in some cases the wrong Arabic words replaced the correct English words. This happens due to similarly pronounced words between the English and Arabic language, so we need to find a way to better recognize which words were actually spoken and eliminate the mistakenly recognized words.

Also this research didn't handle matching notes to slides with only images, so finding a way to handle that case would even increase the potential of using this system on the production level.

Bibliography

[1] Crystal, David (2010). The Cambridge encyclopedia of language (3rd ed.). Cambridge: Cambridge University Press. pp. 374–375.

[2] Li Wei, ed. (2000). The Bilingualism Reader. London: Routledge.

[3] Myers-Scotton, Carol (1989). "Codeswitching with English: types of switching, types of communities". World Englishes. 8 (3): 333–346. doi:10.1111/j.1467-971X.1989.tb00673.x.

[4] Juang, Biing-Hwang, and Lawrence R. Rabiner. "Automatic speech recognition–a brief history of the technology development." Georgia Institute of Technology. Atlanta Rutgers University and the University of California. Santa Barbara 1 (2005): 67.

[5] Ault, Shaun V., et al. "On speech recognition algorithms." International Journal of Machine Learning and Computing 8.6 (2018): 518-523.

[6] *Jo, Taeho. Machine Learning Foundations: Supervised, Unsupervised, and Advanced Learning. Springer Nature, 2021.*

[7] deep learning images. https://developer.ibm.com/articles/an-introduction-to-deep-learning.

[8] Christian Janiesch, Patrick Zschech, and Kai Heinrich. Machine learning and deep learning. 2021.

[9] pdf.js. https://mozilla.github.io/pdf.js.

[10] recorder.js. https://github.com/addpipe/simple-recorderjs-demo.

[11] deep learning model. https://huggingface.co/Pyjay/sentence-transformers-multilingual-snli-v2-500k.

[12] sentence-transformers. https://github.com/UKPLab/sentence-transformers.