CODE-SWITCHING CLASSIFICATION ON HINGLISH

by

Angus Tsang

May 2026

A Technical Report submitted to the
Faculty of the Graduate School of
the University at Buffalo, State University of New York
in partial fulfilment of the requirements for the
degree of

Master of Science

Department of Computer Science and Engineering

Table of Contents

List of Tables

List of Figures

Abstract

The development of the transformer has pushed the development of Automatic speech recognition(ASR) models, greatly improving previous designs in efficiency, speed, and accuracy. However, these same models start to struggle as we step into the realm of multilingual systems. Code-switched audio, the practice of alternating between two or more languages, dialects, or linguistic styles within the same conversation ,serves as one of the defining roadblocks in the development of multilingual ASR models. In this report, we attempt to develop a pipeline to help perform code-switched ASR utilizing a MLP classifier with a Wav2vec head to help identify code-switch boundaries. While the current pipeline does not work as well as hoped, the classifier shows promise as the classifier shows signs of being able to detect the boundary, performing better than random probabilistic chance. However, more work is required to confirm the effectiveness of the classifier head.

Chapter 1

Introduction

1.1 Background

With advancements in machine learning, namely the invention of the Transformer architecture (**?**), speech processing has undergone a paradigm shift. Transformer models have improved the modeling of long-range temporal dependencies and contextual information in speech signals. Moreover, the gains of Transformer-based models have greatly overshadowed traditional RNN models (**?**), enabling the development of highly accurate automatic speech recognition (ASR) systems that still take advantage of neural network architectures while being more computationally efficient. This capstone project aimed to leverage these advances to improve ASR in multilingual and code-switching linguistic contexts. These contexts are especially challenging for ASR because they effectively double the number of possible symbols that the systems must entertain. Being able to anticipate what language will be used when, therefore, could make multilingual ASR systems more efficient and more accurate.

Even cutting edge ASR systems struggle with multilingual and code-switching.

Code-switching refers to the phenomenon where speakers alternate between two or more languages within a conversation or even within a single sentence Such linguistic behavior is common in multilingual communities and social settings, particularly in regions where bilingual or multilingual speakers frequently interact. Unfortunately, traditional ASR systems are typically trained on monolingual data, and thus often struggle to accurately recognize code-switched speech due to differences in phonetics, syntax, pronunciation, and vocabulary across languages. Many papers have attempted to solve this problem (**??**). For instance, Garg et al. (**?**) argued that constructing a "dual" language model could mitigate against poor performance. However, even these solutions have typically lower accuracies compared to monolingual language data because the data are considerably more complex.

Our objective is to develop a system capable of accurately recognizing mixed-language speech while maintaining robustness across different speakers and speaking styles. The current work aims to take advantage of recent self-supervised learning approaches, such as Wav2vec 2.0 (**?**) and Whisper (**?**), which have further improved performance in complex linguistic scenarios. By utilizing large-scale unlabeled audio corpora to learn robust speech representations prior to downstream fine-tuning, models like Wav2vec 2.0[1] and Whisper are able to greatly reduce the need for labeled training data.

1.2 Motivation

Automatic Speech Recognition (ASR) has become an increasingly important field of research due to its broad range of applications, including virtual as-

[1]Wav2vec will be used to refer to Wav2vec 2.0

sistants, accessibility technologies, automated transcription services, and multilingual communication systems. Recent advances in deep learning have significantly improved the performance of ASR systems for monolingual speech. However, accurately transcribing code-switching speech remains a major challenge for current ASR models. As multilingual communication becomes more prevalent, there is a growing need for ASR systems that can robustly handle code-switched speech.

1.3 Goals

The objectives of this project are summarized as follows:

- Develop and train an end-to-end deep learning model for ASR, which takes in raw audio as input and generates the correct textual transcription.
- Reduce transcription errors, including substitutions, insertions, and deletions.
- Improve the model's ability to generalize across different speakers, accents, and speaking styles.
- Evaluate system performance using standard ASR metrics such as Word Error Rate (WER).

1.4 Challenges

Code-switching ASR presents several unique challenges that are not commonly encountered in monolingual speech recognition systems. One of the primary difficulties arises from abrupt language transitions within the same utterance, requiring the model to dynamically identify and adapt to multiple languages in real time. These transitions often occur without explicit cues, making language boundary detection difficult.

Additionally, languages involved in code-switching may contain phonetically similar words with different meanings, increasing the likelihood of recognition errors. Differences in syntax, pronunciation, and sentence structure across languages further complicate the transcription process. Another major limitation is the scarcity of large-scale, high-quality code-switching datasets, as most publicly available ASR corpora are predominantly monolingual.

Environmental factors such as background noise, speaker variability, accent differences, and inconsistent speaking rates also contribute to degraded recognition performance. Together, these challenges make code-switching ASR a complex yet important research problem within modern speech recognition.

However, the introduction of Wav2vec, it may be capable to distinguish and separate features and use them to identify code-switch boundaries. Thus, we've designed a pipeline utilizing Wav2vec to hopefully work on code-switching audio.

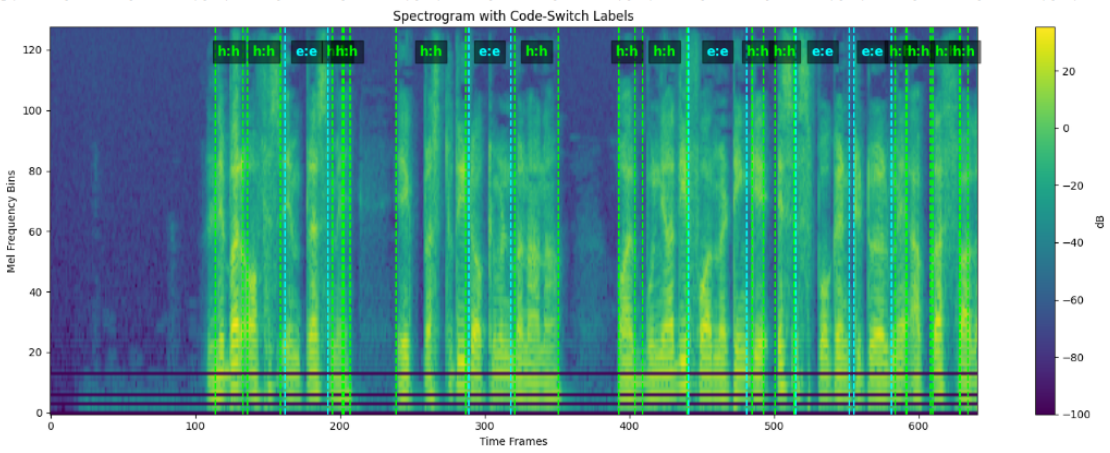

Figure 1.1: Spectrogram with word categories

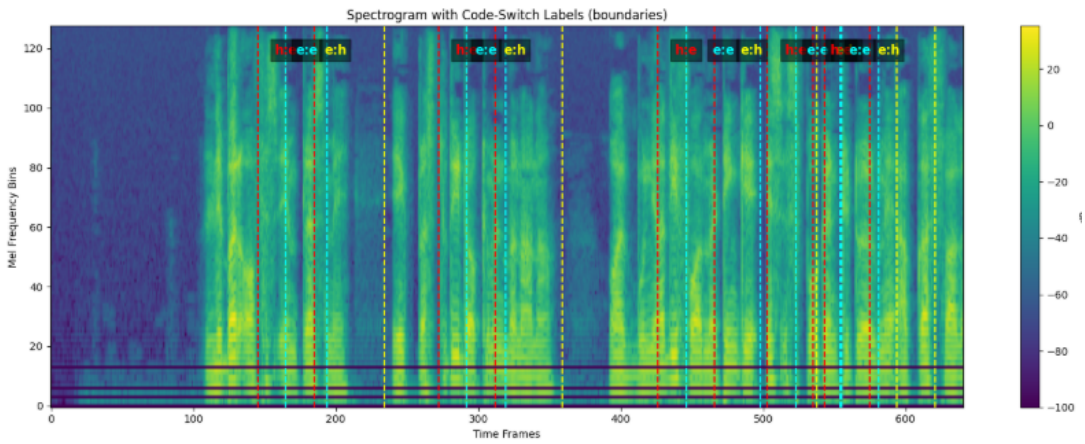

Figure 1.2: Spectrogram with code-switch boundaries

Chapter 2

Methodology

2.1 Model Design

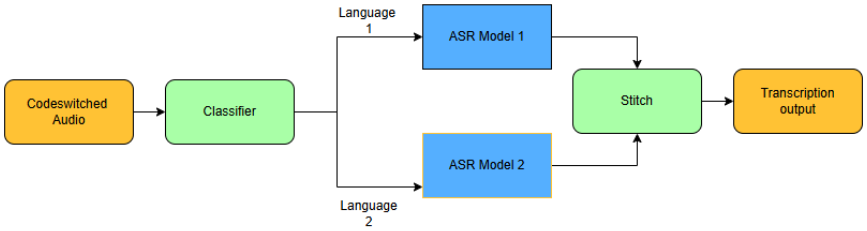

Figure 2.1: General overview of Pipeline Design.

Figure 2.1 shows us the general outline of our ASR pipeline. by utilizing a Classifier to detect the boundaries where the speaker performs a code switch, we hope to reduce the need of more complex ASR models. Our input is an audio segment containing code-switching elements. Using our classifier, we perform a linear scan through the audio in an attempt to find code-switching boundaries. Upon finding a code-switch boundary, we create a new audio segment containing $[timestart : boundary]$. The class outputted by the classifier should allow us to allocate the segments based on the predicted language, which we then use

to pass the new segment into the respective mono-lingual ASR model. Once both ASR models have finished transcribing all their sentences, we can stitch the transcriptions back together, The resulting transcription should provide a complete transcription of the audio.

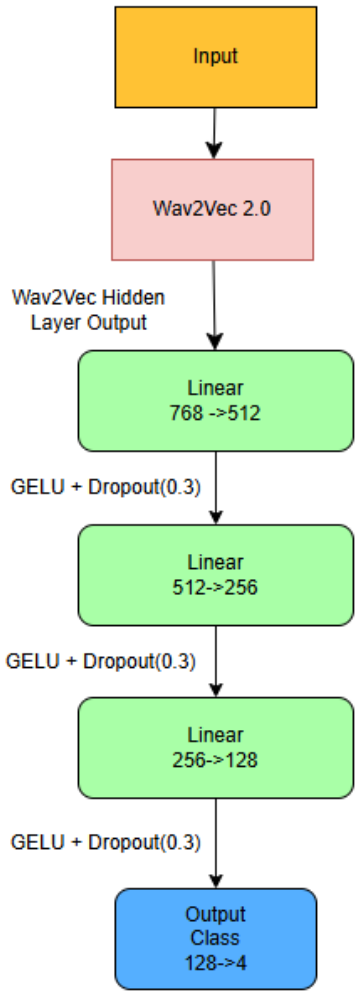

Figure 2.2: Classifier Design.

Figure 2.2 shows the construction of our classifier. We first pass our audio segment through Wav2vec to help extract important features regarding our audio segment. Due to Wav2vec's effectiveness in feature extraction (**?**), the rest

of our classifier is made up of feed forward layers to help retain features that might help us define the code-switching boundary. Between each hidden layer we add dropout to help prevent overfitting. We opt to utilize GELU over ReLU as GELU has typically performed better for deep learning applications (**?**). The classifier outputs one of 4 classes (h:h,e:e,e:h,h:e). where h:h and e:e indicate Hindi and english respectively, and e:h and h:e indicate a switch from the first language to the second language.

Since the focus of this project is on whether the classifier is able to properly identify code-switch boundaries within code-switched audio, we opted to use Whispers pretrained ASR model(small)(**?**) for both ASR models. Depending on the output of the classifier, we would force the ASR model to recognize the segment as the corresponding language, thereby serving as two monolingual ASR models for our pipeline.

2.2 Dataset

We will be utilizing the MUCS 2021 dataset (**?**)[1] for training and testing our ASR pipeline. The MUCS Hindi-English dataset is extracted from spoken tutorials, which cover a wide variety of technical topics and in which code-switching typically comes from the technical content of the lectures (examples would be brands or programs, such as GNU or Linux). The Hindi-English train and test datasets contain 89.86 hours and 5.18 hours of speech audio respectively. However, due to the utilization of Wav2vec, which optimizes and shortens the number of data-points needed for training, We will be using only the Hindi-English test dataset. Additional information regarding the MUCS test dataset is shown

[1]MUCS and MUCS 2021 will be used interchangeably to refer to the dataset

in table 2.1

MUCS Dataset Statistics	
Number of speakers	30
Audio type	Monologue (tutorial)
Number of data points (segments)	3136
% of segments with switching	81.19%
% of segments completely EN	1.37%
% of segments completely HI	17.44%
average % English	25.36%
average % Hindi	74.64%
# of unique English tokens	1293
# of unique Hindi tokens	2622

Table 2.1: MUCS 2021 Hindi-English test dataset

With the unique construction of our pipeline, a couple of changes where made within the Dataset to help facilitate training of the Classifier head. Since our classifier head is only taking in audio slices (which are significantly smaller than the audio segments provided) we opt to split the data provided into word sized audio slices. While the MUCS dataset was originally designed for unlabeled training, to reduce the number of samples needed for training, we label all the slices we will be using for training and testing, these where labeled as either h:h or e:e for Hindi words and english words respectively. However, the current dataset does not have any audio slices containing code-switch boundaries to help train our model to recognize them. So in addition to the original slices, on every occurrence of code-switching we obtain two slices, the beginning boundary and the ending boundary each labeled as either e:h or h:e. we opt to distinguish the two as it is likely that there are distinct features between the two occurrences that the model may be able to distinguish. the resulting set is then split for testing and training. Due to MUCS being heavily imbalanced towards Hindi samples, we utilize a weighted sampler during training to help

overcome sampling bias.

2.3 Training

We trained the classifier on the audio slices that we generated from the MUCS dataset. table 2.2 shows the Hyperparameters used during training the classifier. to help with the class disparity, we also added weights to the loss function to help lower the importance it places on Hindi samples and hopefully focus more on boundary based samples.

Hyperparameter	Value	Notes
Loss	Cross Entropy Loss	added weights
Optimizer	AdamW	
Number of Epochs	25	
Batch Size	8	

Table 2.2: Hyperparameters for Classifier Training

Since the ASR Model is pretrained, we do not need to train the model.

2.4 Testing

We will test the pipeline through 2 different test. The first test will test classifiers accuracy on detecting code-switch boundaries.

The second test will check the pipelines capability to generate correct transcripts. We will pass audio segments to the pipeline and generate complete transcripts from the audio. We will then compare the results to a single pass done with Whisper small on the same test audio as our base case. Sentence correctness will be measured using Word Error Rate (WER) , the standard metric used to measure the accuracy of Automatic Speech Recognition (ASR) systems.

Chapter 3

Results

3.1 Classifier

label	precision	recall	f1-score	support
e:e	36%	52%	42%	1879
h:h	84%	67%	74%	5643
h:e	41%	26%	32%	1879
e:h	51%	77%	61%	1879
macro average	53%	56%	53%	11280
weighted average	63%	59%	60%	11280
Overall Accuracy	59%			

Table 3.1: Classification details

The table 3.1 shows the results after running the Classifier on test data generated from the MUCS dataset. The table shows an overall accuracy of 59%, which indicates that the model is performing significant predictions regarding the class of the audio slice. We can see that our model is fitting to labels h:h and e:h pretty well, with label h:h having a precision of 84% and a recall of 67% indicates that the model has fit pretty well to matching labels h:h and e:h. The difference between the weighted average(60%) and the macro average(53%) in-

dicate a class imbalance between classes, and e dominant class(Hindi) is likely affecting the overall performance of the model.

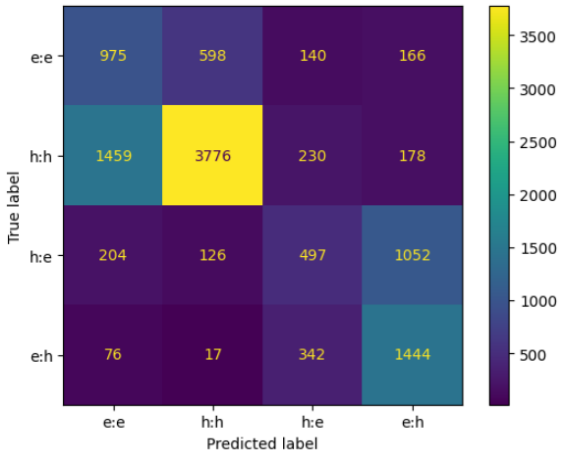

Figure 3.1: Confusion matrix on test data

The confusion matrix shown in figure 3.1, from the confusion matrix, it indicates that the classifier oftentimes confuses oftentimes e:e with h:h and h:e with e:h. This distinction would indicate that switching between two languages contains distinct features that may allow the model to distinguish between the two circumstances. However, this may also be a side affect from the method of processing performed for each label. Since both labels h:e and e:h capture the moment of switching (ie in between words) there would be a distinct gap in each audio vector containing a boundary.

3.2 Pipeline

our pipeline	whisper model(small)
1.47	0.83

Table 3.2: Average WER between pipeline, and whisper model(small)

Table 3.2 shows the difference in WER between our designed pipeline and a single pass with Open-AI's whisper model(small). Both models perform terribly in regards to transcribing the code-switched audio. However, we notice that the mono-lingual ASR model actually does better than our pipeline. Whispers singular model obtains a score of 0.8, which indicates that 4 out of 5 words from the generated transcription do not match with the ground truth.

Our pipeline actually has a WER greater than 1, indicating that our model is likely adding extra words to the transcript. We can see that this is true from the provided example (figure 3.2): the pipeline transcript repeatedly repeats the same word over and over again during transcription. This is likely due to how short each segment provided is compared to typical ranges for ASR. With short transcription windows like we are currently performing, the ASR model does not have enough contextual information to infer the correct text, resulting in these hallucinations.

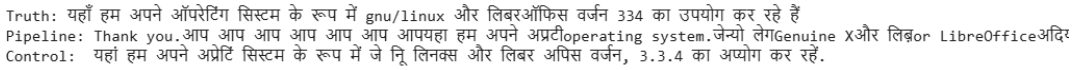

Figure 3.2: Example output of transcripts

Chapter 4

Conclusions and Future Work

4.1 Conclusions

In this work, we attempted to create a new pipeline to help allow whispers multilingual model process and interact with code-switching audio. While the results show that the pipeline struggled (and in fact performed worse compared to just the multilingual model) the classifier head does show signs of fitting and does perform better compared to probabilistic chances. Improvements and modifications to this project may yield better results and allow our pipeline to perform better.

4.2 Future Work

Future work on this project should focus on optimizing and refining the classifier head to help make identifying code-switch boundaries easier. first, due to the simple design of the classifier head, features that may be relevant to identifying code-switching boundaries may not necessarily be detected through the use of Wav2vec 2.0. Improving and refining this classifier head. such as improving the complexity of the classifier layers, or further processing of the audio segment to extract more features that may be relevant to code-switching.

Furthermore, due to restricted resources, we only utilized the test dataset from the MUCS Dataset, Training utilizing the complete MUCS training dataset may help the pipeline learn more distinct features that occur during code-switching and overall increase the accuracy of this pipeline. MUCS is also a very skewed Hindi dataset, even with the addition of weighted sampling, our data indicates that the model is still being skewed based on the dominant class. Weighted sampling also introduces the problem of overfitting due to lack of english segments, making it insufficient to solve the class disparity. Including a more english balanced dataset into the equation would help balance out these problems. Furthermore, diversifying languages may similarly present unique features that are unexplored when performing code-switching between Hindi and english. Thus, including such corpora(**??**) may help develop the model even further.

Lastly, current designs of mono-lingual ASR are not fit to transcribing small audio segments. Such instances of short burst of languages naturally occur during code-switching, and thus make current ASR models difficult to implement into this pipeline. Focus on creating different, more specialized ASR models for small audio segments will help decrease the WER during transcription.