Project Report: Forced Alignment using Montreal Forced Aligner (MFA)

Name: Chinimilli Hari Prasad

Email: hariprasadchinimilli18@gmail.com

Phone: 9346416669

GitHubRepository: https://github.com/Hari-1718/MFA-Forced-Alignment-IIITH

1. Introduction

For this assignment, I had to build a pipeline to automatically match audio recordings with their text transcripts. This process, called "Forced Alignment," helps us find the exact start and end times for every word and every individual sound (phoneme) in a speech file. I used the Montreal Forced Aligner (MFA) to get this done.

2. My Step-by-Step Process

Step 1: Setting up the environment

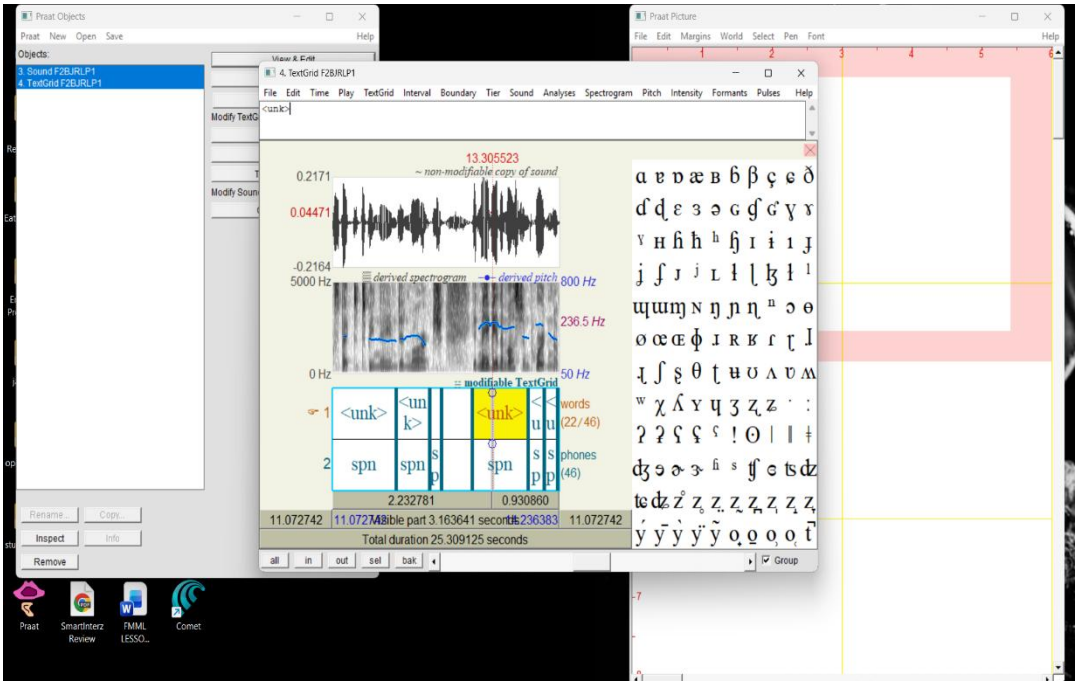
First, I set up a clean environment using Conda to make sure all the tools worked correctly. I installed the montreal-forced-aligner and its dependencies like Kaldi.

Step 2: Preparing the Data

The audio files were in a wav folder and transcripts were in a transcripts folder. MFA needs them together with matching names. To save time and avoid mistakes, I wrote a Python script called prepare_data.py to automatically pair them up into a new folder called mfa_data.

Step 3: Initial Alignment & OOV Problem

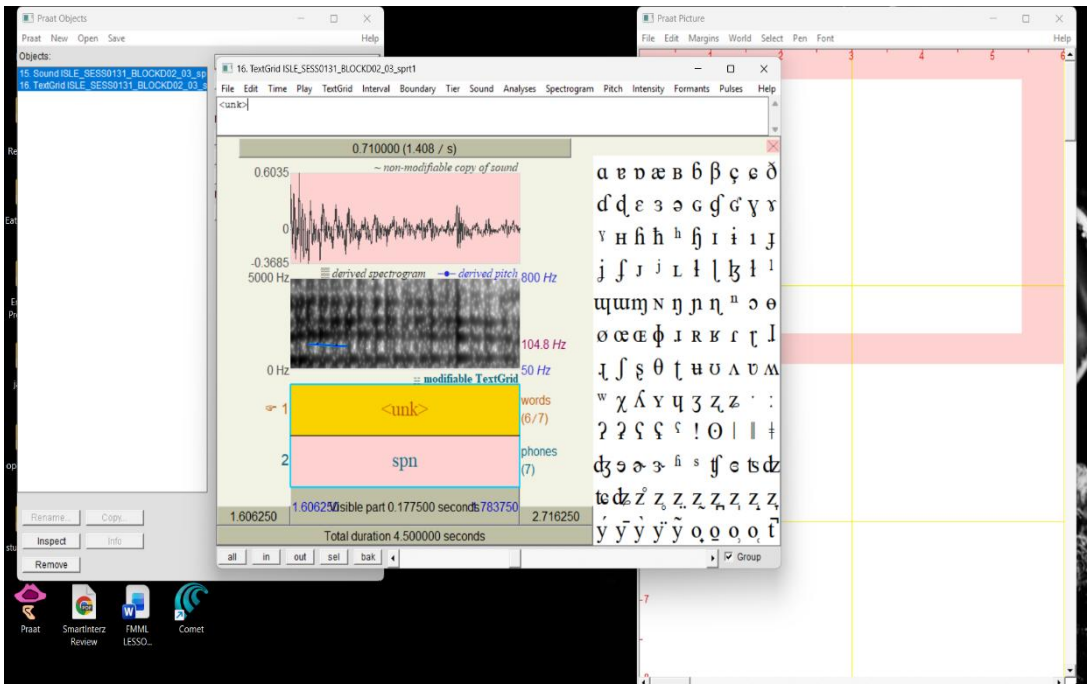
I used the standard english_us_arpa acoustic model and dictionary for the first run. However, I noticed that several words were missing from the standard dictionary. These are called Out-Of-Vocabulary (OOV) words. In my case, I found **17 OOV word types**, mostly names like "Dukakis" and "Hennessy."



This shows the initial run where words like 'Dukakis' were unknown (<unk>), causing gaps in the alignment.

Step 4: Fixing the OOV Words (G2P Model)

To fix the gaps, I used a G2P (Grapheme-to-Phoneme) model. This tool "guesses" how a word should be pronounced based on its spelling. I generated a new file called custom_dict.dict that included these missing pronunciations.



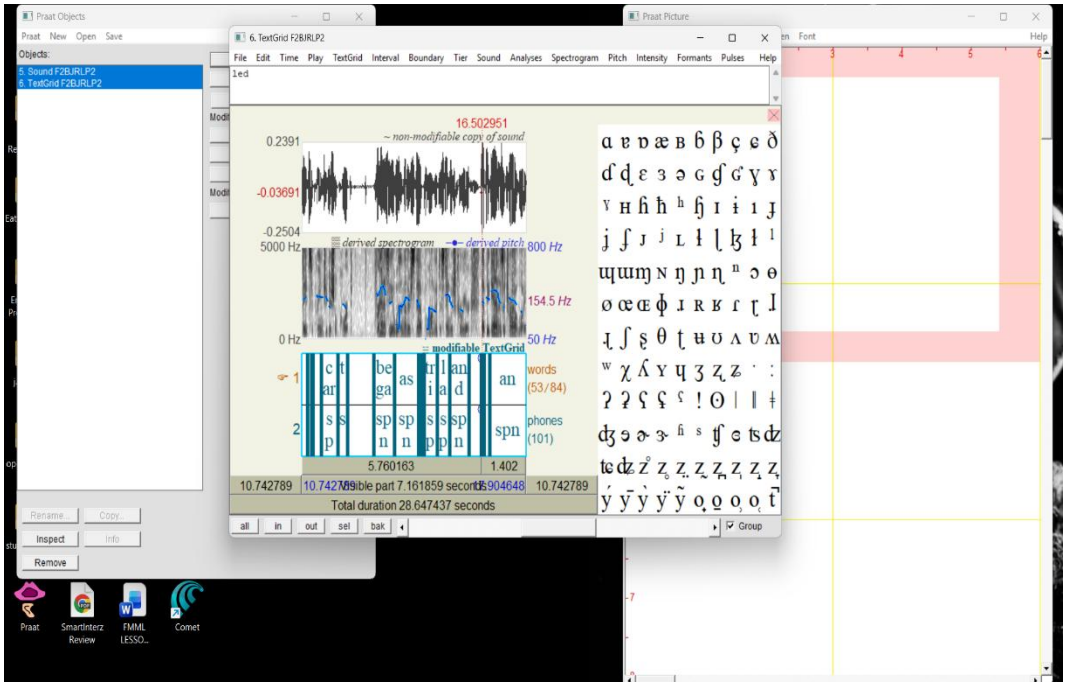
Step 5: Final Alignment

I ran the aligner again, but this time I used my new custom dictionary. This filled in all the gaps. I then exported the results as TextGrid files.

3. Inspecting the Results in Praat

I opened the final .wav and .TextGrid files in Praat to see if the lines actually matched the voice.

- **Accuracy:** The boundaries for the words and individual sounds matched up very well with the "spikes" in the audio waveform.
- **Pauses:** I noticed that MFA is smart enough to mark silences or background noises as <vsp> (voice speech pause) so they don't mess up the word timings.

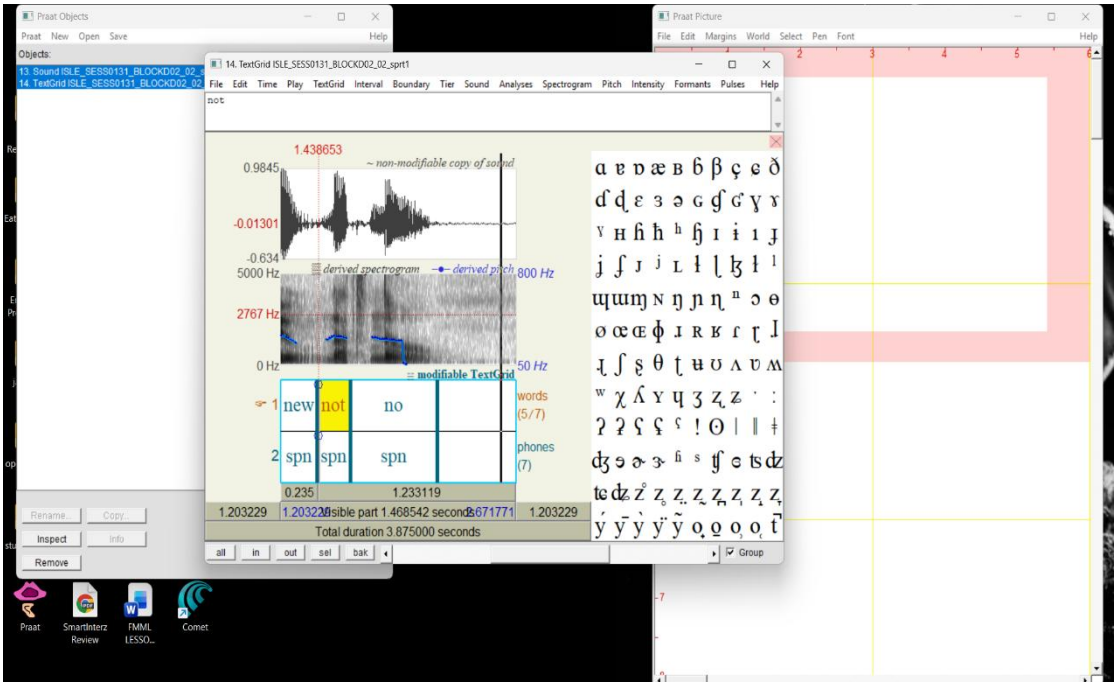


A zoomed-in view in Praat showing how the word and phone boundaries now align perfectly with the audio waves.

4. Key Observations

1. **OOV Handling is Crucial:** Without using the G2P model, we lose a lot of important data (like names). Adding a custom dictionary made the alignment complete.

2. **Boundary Mismatches:** Sometimes there is a tiny delay if the speaker breathes or makes a small noise before speaking, but overall, the MFA boundaries are very precise.



3. **Visualization:** Looking at the spectrogram and waveform together in Praat really helps confirm that the "p" or "s" sounds are being cut at the right millisecond.

5. Final Repository Structure

My GitHub repository is organized like this:

- mfa_data/: My prepared input files.
- mfa_outputs_final/: The final TextGrid results.
- alignment_visualizations/: Screenshots showing my work.
- prepare_data.py: The script I used for data prep.
- custom_dict.dict: The solution I created for the missing words.