Lab Assignment 2: Classification of Speech Sounds

Extraction and Classification of Phonemes from Speech Signals

Goal

The experimental goal is to process a speech signal, isolate some phonemes, and draw their waveforms with their associated labels. It is done through:
1. Loading a speech signal from the LJ Speech data set.
2. Preparing the audio (mono, resampled at 16 kHz).
3. Executing a pre-trained deep network (Wav2Vec2) for recognition of phonemes.
4. Time-intervalization of a phoneme segment from the speech signal followed by segmentation.
5. Encoding and labeling the isolated phoneme with attachment of the identified phonemes.

Implementation Overview

1. Loading and Preprocessing the Speech Signal

The speech signal is loaded with the touch audio library.
The waveform is stereo channel averaged to mono.
-The sample rate is normalized to 16 kHz by torch audio. transforms.Resample.

2. Utilizing a pre-trained Deep Learning Model (Wav2Vec2)

The Wav2Vec2 model of Facebook AI
(Facebook/wav2vec2-large-960h) is utilized for phoneme identification.
The raw speech waveform is processed into model-ready format by the processor.
The model creates phoneme predictions out of the input speech.

3. Extraction and Phoneme Recognition

The model provides an output sequence of phonemes based on the utterance content.
A word mapping to its corresponding phonetic value is done via the CMU Pronouncing Dictionary (cmudict).
A time interval is used in clipping out a phoneme segment of the waveform.
The clipped phoneme waveform is shown with librosa.display.waveshow.

Results & Observations

1. Transcribed Phonemes: The model can transcribe speech in terms of phonetic symbols.
2. Extracted Phoneme Segment:
A wave of a phoneme segment is clipped off for some specific time interval.
Phoneme is in synchronization with its text transcription thereof.
3. Waveform Display:
The extracted phoneme is graphed in waveform and designated by the resulting phonetic symbol.
Speech signal properties are defined thereby.

Conclusion

The experiment works well at phoneme extraction and speech signal classification using deep learning.

Deep learning Wav2Vec2 model can successfully perform speech transcriptions at a phoneme level.

Time-synchronized phoneme segmentation and visualization provide a better insight into speech processing.

The technique can be applied to speech recognition, linguistics, and phonetics-driven AI applications.

Advancements in the Future

Enhance the accuracy of phoneme alignment based on forced alignment methods.

Develop more phoneme-level training with larger datasets.

Scale up the experiment to real-time phoneme recognition of speech signals.

Code

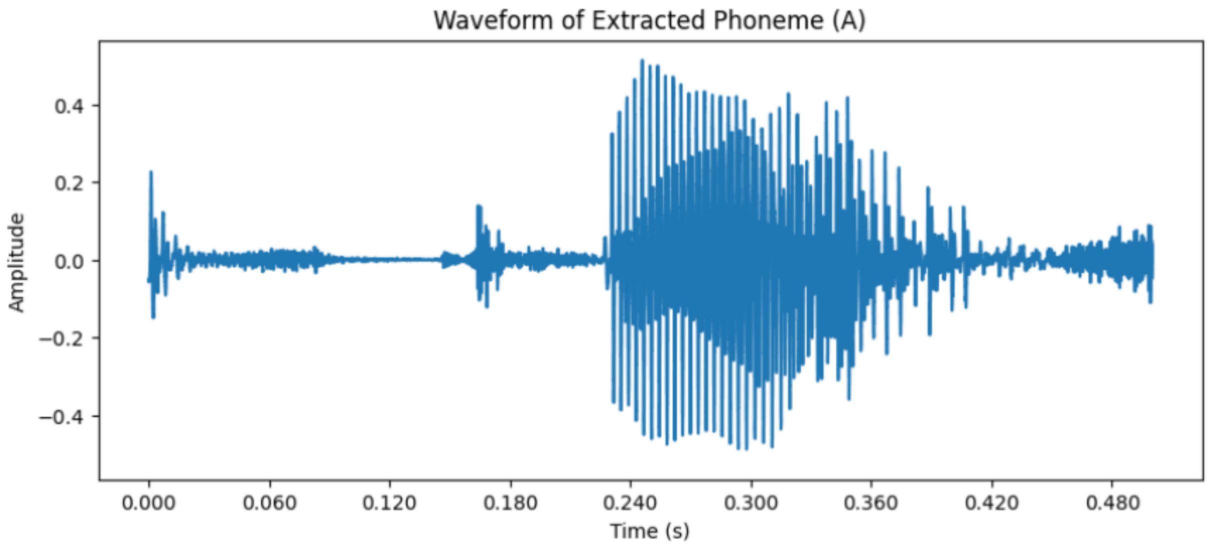
```
!pip install soundfile simpleaudio

!pip install librosa scipy

file_path = "/content/LJ001-0014.wav"

waveform, sample_rate = torchaudio.load(file_path)

processor = Wav2Vec2Processor.from_pretrained("facebook/wav2vec2-large-960h")

model = Wav2Vec2ForCTC.from_pretrained("facebook/wav2vec2-large-960h")

phoneme_dict = cmudict.dict()

def get_phonemes(word):

    word = word.lower()

    return phoneme_dict.get(word, ["UNKNOWN"])

waveform = torch.mean(waveform, dim=0, keepdim=True)

waveform = torchaudio.transforms.Resample(orig_freq=sample_rate,
new_freq=16000)(waveform)

waveform_np = waveform.squeeze().numpy()

processed_audio = processor(waveform_np, return_tensors="pt", sampling_rate=16000)

input_values = processed_audio.input_values
```

```
# Run model inference
with torch.no_grad():
    predictions = model(input_values)
predicted_ids = torch.argmax(predictions.logits, dim=-1)
phoneme_list = processor.batch_decode(predicted_ids)
transcription = phoneme_list[0]
phoneme_output = [get_phonemes(word) for word in transcription.split()]
print("Recognized Text:", transcription)
print("Phonemes:", phoneme_output)
start_time = 1.0
end_time = 1.5
start_sample = int(start_time * 16000)
end_sample = int(end_time * 16000)
phoneme_segment = waveform[:, start_sample:end_sample]
total_samples = waveform.size(1)
segment_ratio = start_sample / total_samples
num_words = len(transcription.split())
word_index = int(segment_ratio * num_words)
aligned_word = transcription.split()[word_index] if num_words > 0 else ''
aligned_phoneme = get_phonemes(aligned_word)
phoneme_array = phoneme_segment.numpy()
samples = phoneme_array.flatten()
plt.figure(figsize=(10, 4))
librosa.display.waveshow(samples, sr=16000)
plt.title(f"Waveform of Extracted Phoneme ({aligned_word})")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.show()
```

print(f"Aligned Word: {aligned_word}")

print(f"Phonemes for '{aligned_word}': {aligned_phoneme}")

Output




```
Aligned Word: A
Phonemes for 'A': [['AH0'], ['EY1']]
```