Training Report - By Gorden Lim

Observations comparing transcribed text

- Custom Pretrained model has blanks returned for certain audio files
- Custom Pretrained model seems to be poorer at detecting spaces
- A lot more ‘nonsense’ words are made up

1) Blank transcribed audios

The first thing that immediately stands out between my results and the original fine-tuned wav2vec2 model is that my fine-tuned model has a lot of predictions where <blank> is returned. This could be due to the much shortened training dataset, and the fact that the training loss does not seem to converge during training, leading the model to underfit and thus product default to blanks for some inferences.
Using the full common voice dataset, and the inclusion of other datasets, as well as more epochs during training should prevent this issue from recurring.

2) Poor <pad> token detection

Another observation is my pre-trained model is poorer at detected spaces (or gaps between words). This could be due to the lack of data again. Although it could be worth investigating if some simple post-processing (e.g fuzzy matching against a dictionary) could alleviate this issue and improve the WER.

3) Incorrectly spelt words

Lastly, I also noticed my pre-trained model has a lot of ‘phonetically correct’ but wrongly spelt words. For example, ‘xylophone’ becomes ‘cilepone’. This could be due to the limited vocabulary coverage in the training data, and rarer words such as xylophone are never learnt by the model and thus the correct spelling is not known. We could make use of a Language Model for decoding (making use of HuggingFace Wav2Wev2ProcessorWithLM). We can also augment our data by compiling all ‘rare’ words within our dataset to identify which common words which might be more scarce in our dataset.

Other experiments I would try

1) Training without Gradient Accumulation

   I used gradient accumulation during training to simulate a larger batch in order to reduce RAM usage, but for the purposes of finetuning, it might be better to use smaller batches that fit in memory.

2) Training with a Language Model

Came across this blogpost about fine-tuning Wav2Vec2 with LM (link). The process aims to improve the decoder output with an external n-gram Language Model. HuggingFace also has processors for Wav2Vec2ProcessorWithLM.