Steven Rud and Daniel Mints COSI 136a

Training and evaluating an ASR system Report

In our code we implemented an Automatic Speech Recognition (ASR) system on the Ukrainian language using a combination of Wav2Vec2+CTC model and medium-sized and small-sized Whisper models. As our corpus we used the Mozilla Common Voice (Common Voice Corpus 5.1). We used the pretrained medium and small Whisper models as our angle to compare how and why each model is as effective as it is. The ASR system is trained on a dataset and then tested on a separate dataset for evaluation.

Training and creating the model

So, before we begin our analysis, we firstly need to train a model. To do this we firstly load in the data of 250 audio files, having a rough duration of 21 minutes and 3 seconds (past the minimum requirement) from the train.tsv file in the corpus. And then we initialize our Wav2Vec2+CTC architecture using the ‘facebook/wav2vec2-base-960h’ directory and we used an AutoTokenizer from Hugging Face's Transformers library to obtain an appropriate text tokenizer. Next, we prepare the audio files using the ASRDataset class. This class handles the dataset by loading the audio files, resampling them to a common sample rate of 16000 Hz, ensuring a consistent waveform length by padding or truncating to a desired length of 156672, and returning the waveform along with the corresponding correct given transcript from the dataset. Now we can run the audio through a training loop, where the model is trained using the CTC loss. The training loop runs for four epochs. I found four epochs to be the most ideal as

with any more epochs the total loss begins to stagnate or actually increase. Here is the average loss per epoch:

Epoch 1, Average Loss: 11.092069871841915

Epoch 2, Average Loss: 6.591654640341562

Epoch 3, Average Loss: 5.9649963406106785

Epoch 4, Average Loss: 5.124443820544651

During each iteration, the model processes batches of audio waveforms and their corresponding transcripts, calculates the CTC loss, and updates the model's parameters using the AdamW optimizer. Lastly, the trained Wav2Vec2+CTC model and tokenizer are saved for later use.

Now for a quick more technical overview. The Wav2Vec2+CTC Model Utilizes a convolutional neural network (CNN) for feature extraction and employs context-to-context (C2C) self-supervised learning for pre-training. It also uses Connectionist Temporal Classification (CTC) loss for training and the tokenizer converts transcripts to input IDs for the CTC loss calculation. To quickly cover the hyperparameters in the training loop, we use as previously mentioned an AdamW optimizer with a learning rate of 1e-4 and 4 epochs.

Evaluating an ASR system

Firstly, we again use the ASRDataset to prepare and ensure that the files are prepared. We again load the audio files, resample them to a common sample rate of 16000 Hz, ensure a consistent waveform length by padding or truncating to a desired length of 156672, and return the waveform along with the corresponding correct given transcript from the datasets. The data is

loaded from the Common Voice Corpus with test.tsv as the baseline and dev.tsv as the comparison partition for the data. The models we loaded are the previously made Wav2Vec2+CTC model and tokenizer and the medium-sized and small-sized Whisper model from the Whisper library. I again use 250 audio files for the data processing and evaluating. Now getting into the actual processing for each dataset I use both the medium Whisper model and the Wav2Vec2+CTC model to transcribe the audio. The results are combined, preprocessed, and compared with the correct transcript to calculate Word Error Rate (WER) and Character Error Rate (CER). The preprocessing includes lowercasing the texts and removing punctation to ensure capitalization and punctation doesn't lead to any inaccuracies, as that isn't the focus of the ASR system. I also made sure the Cyrillic "і" is used in the datasets as I found that the data occasionally used the Latin "i". I also utilized the CER calculation to ensure if there were any very minor typos with the transcription it would still consider it correctly transcribed. Lastly, the code accumulates statistics such as transcription comparisons, WER, and CER for further analysis. It also prints and stores information for both the test and dev dataset. This is all saved onto four different CSV files, which are provided in the GitHub repository. Each CSV has a version of the models and dataset, with it being run on test.tsv on a medium whisper model, on test.tsv on a small whisper model, on dev.tsv on a medium whisper model, and on dev.tsv on a small whisper model.

Data Description before Result Analysis

Now before we get into the results analysis, it would first be important to understand the data we used. The Ukrainian Common Voice Corpus 5.1 was made on 7/13/2020 with 26 recorded hours, 23 validated hours, and 235 different voices. The created Wav2Vec+CTC model was as

previously mentioned based on Facebook's Wav2Vec+CTC model. The pretrained Whisper models are downloaded from openAI. This model is a Transformer based encoder-decoder model, also referred to as a sequence-to-sequence model, trained on 680k hours of labelled speech data annotated using large-scale weak supervision. The multilingual models were trained in both speech recognition and speech translation. For speech recognition, the model predicts transcriptions in the same language as the audio, while for speech translation, it anticipates transcriptions in a different language than the audio. Now moving onto our medium and small sized models. Their primary difference was the amount of parameters they were trained on, with the medium model being trained on 769 M parameters and the small model being trained on 244 M parameters.

Results and Quantitative and Qualitative Analysis

Firstly, all the results I got, more details for each audio file are provided in the respective CSV files. The Average CER was calculated by dividing the total CER by the total word count. The Average WER was calculated by dividing the total WER by the total number of audio files. The overall WER and CER were calculated by using the 'wer' and 'cer' functions from the 'jiwer' libraries on all the correct transcriptions and all my transcriptions created from the combined Wav2Vec+CTC and whispers models.

Results:

When Ran on test.tsv on a small whisper model:

Average Character Error Rate (CER): 0.0023867604259088598 **(0.24%)**

Average Word Error Rate (WER): 0.3692368187368188 **(36.9%)**

Overall Character Error Rate (CER): 0.10069241678485026 **(10.1%)**

Overall Word Error Rate (WER): 0.34573894282632145 **(34.6%)**

Total correct transcriptions: 152 out of 250 **(60.8%)**

When Ran on test.tsv on a medium whisper model:

Average Character Error Rate (CER): 0.001478902005696787 **(0.15%)**

Average Word Error Rate (WER): 0.22944335664335677 **(22.9%)**

Overall Character Error Rate (CER): 0.0594811045299074 **(5.95%)**

Overall Word Error Rate (WER): 0.21305285868392665 **(21.3%)**

Total correct transcriptions: 194 out of 250 **(77.6%)**

When Ran on dev.tsv on a small whisper model:

Average Character Error Rate (CER): 0.0021044653870353003 **(0.21%)**

Average Word Error Rate (WER): 0.33650698190698214 **(33.7%)**

Overall Character Error Rate (CER): 0.0898952924150847 **(8.99%)**

Overall Word Error Rate (WER): 0.3290529695024077 **(32.9%)**

Total correct transcriptions: 165 out of 250 **(66.0%)**

When Ran on dev.tsv on a medium whisper model:

Average Character Error Rate (CER): 0.0011722935484108532 **(0.12%)**

Average Word Error Rate (WER): 0.20294115884115888 **(20.3%)**

Overall Character Error Rate (CER): 0.04928918021622542 **(4.93%)**

Overall Word Error Rate (WER): 0.19743178170144463 **(19.7%)**

Total correct transcriptions: 204 out of 250 **(81.6%)**

So now analyzing our angle of comparing the medium and small whisper models, we can see that for both the Average and Overall CER in both the test and dev dataset for the medium sized whisper model is roughly twice as smaller than the CER for the small sized whisper model. And similarly for the Average and Overall WER in both the test and dev dataset the word error rate for the medium sized whisper model is roughly one and a half times smaller than the WER for the small sized whisper model. The Total Correct Transcriptions for both the test and dev dataset is better by roughly 16% when using the medium sized model instead of the small model. So, the medium model consistently outperforms the small model in character and word-level accuracy, and general correct transcriptions. Both models also achieve higher accuracy and lower CER and WER for the dev.tsv dataset compared to the test.tsv dataset. The reason behind the medium sized model consistently performing better than the model is the over 3 times amount of parameters it used when being trained, as shown earlier in the model analysis. So, it was able to capture more complex patterns in the data and better generalize to various accents and speech

patterns, resulting in improved performance. However, due to the increased amount of parameters the run time of the medium model was twice as long as when running the small model. So, in most contexts the error rates being twice as good for the medium model and the higher transcription accuracy percentage corresponds with the extra run time. This suggests that the larger model size and possibly more extensive training data contribute to better ASR performance. Perhaps with a more improved Wav2Vec+CTC model we can improve the smaller whisper model as it would provide the necessary support needed for it to make up for its gaps and help catch up to the medium model in terms of general accuracy with a shorter run time.

GitHub Repository: https://github.com/StevenRud7/COSI136a-Assessment-2