ELEC-E5510 – Speech Recognition

Assignment 5

Nguyen Xuan Binh, 887799

29/11/2023

Question 1

1. Report the WER on the data stored in `pairs_batch_test_clean` and `pairs_batch_test_other`. Use up to two decimal points.

The WER of pairs batch test clean is 0.24

The WER of pairs batch test other is 0.37

2. Why do you think the WER is worse on `pairs_batch_test_other`? Which factors could impact the performance?

“pairs_batch_test_other” dataset has a worse WER than that of “pairs_batch_test_other”.

This is because the original data is trained on a bi-gram language model and they are ranked by the WER. However, the test other dataset has speakers with higher WER, which means the model struggles to recognize the speech from new speakers, possible due to overfitting, or due to the lack of robustness in the model

Factors that could impact the performance are background noise, mispronunciation, or there are many uncommon vocabularies (OOV) in the utterances

Question 2

The variable `pairs_batch_test_long` contains a subset of the clean LibriSpeech test set, where the utterances are longer (longer sentences being spoken).

1. Test the performance of the model on the data stored in `pairs_batch_test_long` (use up to two decimal points). How do long utterances affect the performance of the model?

The WER of “pairs_batch_test_long” is 0.35.

This WER is higher than the WER of the “test clean” dataset but lower than the WER of the ”test other” batch.

Based on the comparison, long utterances seem to decrease the performance of the model because the WER has increased significantly compared to the “test clean” dataset.

2. Why do you think that is the case?

The long and short sentence difference has a significant effect on the training process, which depends on the attention mechanism. In other words, longer contexts are very hard to capture because the attention calculation is computationally extensive for longer sequences.

Additionally, the model is not trained on enough long audios, so it will naturally perform badly on long inputs. Using RNN is also not a viable option because of the vanishing gradient problem when RNN are trained on very long inputs.

Question 3

1. As a decoding strategy, we are using greedy decoding. What are the pros and cons of this?

The pros of greedy decoding:

- Only considers most recent inputs, which reduces model complexity

- has a lower training time

- is less computationally expensive to train

The cons of greedy decoding:

- On longer inputs, the model is suboptimal due to the greedy nature of decoding since many candidates in the input sequence will have very low probability

- This strategy may cause a loss in meaning since it only selects the nodes with the highest conditional probability and simply predicts words in sequence, which may go off the trajectory.

2. Propose a better decoding algorithm that overcomes the downsides of greedy decoding.

There is a better option than the greedy decoding, which is the beam search algorithms

Beam search is a search algorithm that tries to find the most likely sequence of choices (like words in a sentence) by keeping track of a set of the best options at each step instead of just one best option like greedy decoding does. As a result, beam search is more likely to find the most generally suitable output than greedy decoding

for example, can overcome the above-mentioned weakness of greedy encoding. It could track multiple words at every step and use those to generate multiple hypotheses.

We can also use a pure sampling decoder, which is a stochastic approach to sequence generation. Unlike greedy decoding, which always picks the next most likely element, pure sampling randomly selects the next element based on the probability distribution given by the model. This means that sometimes it might choose less likely elements, adding randomness and variety to the generated sequences.

Question 4

In the current implementation of the `Encoder`, we are using a standard BLSTM for processing the input features.

1. What are the issues with using this type of `Encoder`?

- Real-time speech recognition: A BLSTM processes the input data in two directions: one LSTM goes forward through the input sequence, while another goes backward, starting from the end. This means that the BLSTM has access to both past and future context at every point in the sequence. When the BLSTM is used, we can't start processing the input for real-time predictions until the entire sequence has been received, which isn't practical for real-time speech recognition
- Sequential processing that limits parallelism: BLSTM processes data recurrently, meaning that the computation for the next step in a sequence can only begin once the current step is complete. This makes training it very slow compared to Transformers, which can handle multiple sequence elements at the same time independently
- Computational complexity: BLSTM has a very large number of parameters, which is harder to train to obtain a good stable model

2. How can those issues be solved?

- To resolve the issues of real-time speech recognition, we can use a pyramidal BLSTM, which is a modified version of the standard BLSTM designed to reduce the computational complexity and allow for faster processing, which can be particularly useful in tasks like real-time speech recognition. This pyramid structure shortens the sequence length with each successive layer, which reduces the total number of computations required. This makes it a suitable architecture for real-time applications where speed is crucial and some loss of detail is acceptable.

- To enable parallelism, we can resolve it by using Transformers instead of BLSTM models as discussed previously