Assignment 4

Your name: Nicholas Changwoo Park
Your GTID: 903549867

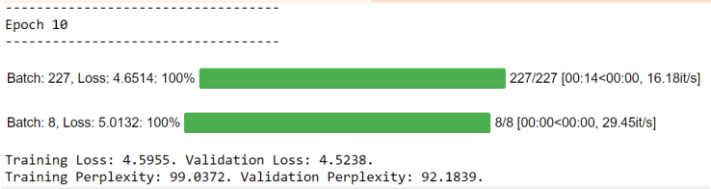
Seq2Seq Tuning Method:

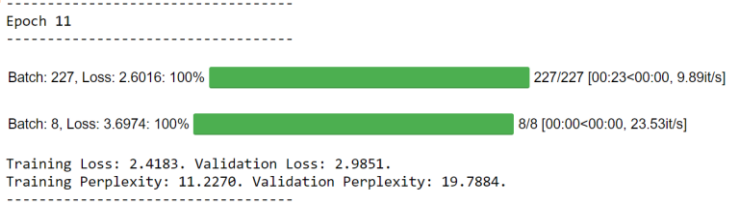
- A grid search was done on {batch_size, learning_rate, encoder_emb_size, encoder_hidden_size, encoder_dropout, decoder_emb_size, decoder_hidden_size, decoder_dropout, model_type, epochs} under an Adam optimizer, over the following discrete points.
  - Batch Size: {16, 32, 64, 128}
  - Learning rate: {1e-4, 1e-3, 1e-2}
  - Encoder/Decoder embedding size: {32, 64, 128, 512}
  - Encoder/Decoder hidden size: {64, 128, 256, 512}
  - Encoder/Decoder dropout probability: {0, 0.2, 0.25, 0.5}
- The Adam optimizer is used (left as default) to incorporate the concept of momentum for a stabler convergence.
- The bidirectional parameter for LSTM and RNN are kept as false. It can further improve seq2seq .
- Supplying the hidden sizes with enough parameters seem to result in an improved performance. It appears that using emb_size too big relative to the hidden_size results in diminished accuracy. The emb_size needed to be tuned with attention to the hidden_size.
- It is possible to overtrain as the epoch increases, noted by the increasing validation loss and perplexity. The weights are likely overfitting to the train data. Epoch is therefore not individually tested as it is possible to implement a method to keep a model with the best validation loss/perplexity achieved so far.

Seq2Seq Tuning Results:

- The search of combination reveals the following tuning achieved the best perplexity and loss.

Before tuning	Loss/Perplexity	After tuning	Loss/Perplexity
Batch size = 128	• Best Validation Loss: 4.52 • Best Validation Perplexity: 92.1839	Batch size = 128	• Best Validation Loss: 2.9851 • Best Validation Perplexity: 19.7884
Learning Rate = 1e-3		Learning Rate = 1e-3	
Decoder/Encoder emb_size = 32		Decoder/Encoder emb_size = 512	
Encoder hidden_size = 64		Encoder hidden_size = 512	
Decoder Hidden_size = 64		Decoder Hidden_size = 512	
Model type = “RNN”		Model type = “LSTM”	
Epochs = 10		Epochs = 11	
Dropout (encoder/decoder) = 0.2/0.2		Dropout (encoder/decoder) = 0.2/0.0	





Seq2Seq Tuning Explanation:

- {epoch, model_type, dropout} are also experimented separately, ceteris paribus, to understand their impacts:
  - More epochs results in improved validation perplexity and loss, but after a certain number of epochs the perplexity and loss are observed to rather increase – i.e. U-shaped curve in performance due to overtraining.
  - LSTM performed nearly twice better than RNN in terms of validation perplexity. This is likely attributed to the additive encoder connection, preventing gradient explosion.
    - Bidirectional LSTM seems to work slightly better than uni-directional with no other parameter tuning likely due to additional context information provided to the network.
  - Regularization: Dropout of 0.2 seems to work well to avoid overfitting to the parameters.
    - It appears the impact of the encoder dropout probability is more significant than that of the decoder. In fact, there was a slight increase in performance when the decoder's dropout = 0 according to the results.
- Batch size that is too large resulted in a decreased performance in terms of validation loss/perplexity. Batch size of 32~256 seemed to be the best performing.
- There are additional parameters to explore beyond what we are tuning, such as num_layer or the choice of nonlinearity in the type or RNN model. Also, with more computation and time, a more detailed grid search (or perhaps with the aid of other search algorithms – e.g. Beam search) is expected to increase the accuracy further.

Transformer Tuning Method:

- The model included the choice to use 4 self-attention heads instead of 2. This is limited to 4 due to the need to implement each head manually in the multi_head_attention() function. Increasing the number of heads initially was thought to improve the final validation loss/perplexity due to more context information fed to the model.
- The model included the choice to use two dropout layers before the two fully connected forward functions (feedforward and final) with a default keep probability of 1 (no dropout).
- Dimensions for the multi-head self-attention k, q and v are kept identical to each other to fit the linear operations consistent to the required inputs and outputs.
- A grid search was done on {batch_size, learning_rate, hidden_dim, dim_feedforward, dim_k, dim_v, dim_q, epochs} over the following discrete points. The choices are accomodated for the GPU's computation ease:
  - Batch_size = {32, 64, 128}
  - Max_len = {20, 64, 128}
  - Num_heads = {2, 4}
  - Dropout probability = {0, 0.1, 0.2}
  - learning_rate = {1e-4. 1e-3, 1e-2}
  - hidden_dim = {128, 256, 512}
  - dim_feedforward = {1024, 2048, 4096}
  - dimkvq = {96, 384, 768}
- To cut scope of hyperparameter space to search within, the optimizer is consistently kept as Adam.
- Epoch is not explicitly searched, due to varying overtraining regions in terms of it.

Transformer Tuning Results:

- The following hyperparameters produced the best validation loss/perplexity so far.

Before tuning	Loss/Perplexity	After tuning	Loss/Perplexity
Batch size = 128	• Best Validation Loss: 2.9924 • Best Validation Perplexity: 19.9342	Batch size = 128	• Best Validation Loss: 2.8600 • Best Validation Perplexity: 17,4609
Learning Rate = 1e-3		Learning Rate = 1e-4	
Hidden dimension = 128		Hidden dimension = 256	
Dim. Feedforward = 2048		Dim. Feedforward = 4096	
Dim. K, V & Q = 96		Dim. K, V & Q = 768	
Epochs = 10		Epochs = 29	
Number of att. heads = 2		Number of att. heads = 4	
Dropout probability = 0		Dropout probability = 0.1	
Max_len = 20		Max_len = 64	

```
---------------------------------
Epoch 10
---------------------------------
Batch: 227, Loss: 2.3675: 100% 227/227 [00:04<00:00, 52.70it/s]
Batch: 8, Loss: 3.7376: 100% 8/8 [00:00<00:00, 100.21it/s]
Training Loss: 2.2826. Validation Loss: 2.9924.
Training Perplexity: 9.8022. Validation Perplexity: 19.9342.
---------------------------------
```

```
---------------------------------
Epoch 29
---------------------------------
Batch: 227, Loss: 2.5675: 100% 227/227 [00:11<00:00, 20.28it/s]
Batch: 8, Loss: 3.5668: 100% 8/8 [00:00<00:00, 52.55it/s]
Training Loss: 2.5783. Validation Loss: 2.8600.
Training Perplexity: 13.1750. Validation Perplexity: 17.4609.
```

Transformer Tuning Explanation:

- Having a big enough sequence length of the tokens resulted in an improvement in accuracy due to more capacity to store word embeddings with granularity – better exploiting the model's capacity. More epochs were needed to reach min. validation loss.
- With all other hyperparameters being equal:
  - Like seq2seq, the validation perplexity/loss exhibits a U-shaped pattern as the epoch increases due to overtraining. This is because, with or without dropout regularization, overfitting on the train set is always a possibility.
  - The model performed the best when the feed forward dimension was the highest. This is the classification layer part of the network that appears to benefits from a sufficient model capacity in terms of parameters.
  - In contrast, increasing the number of self attention dimensions (dim_k/v/q) from 96 did not show significant variation in the best final validation perplexity. Likewise, increasing the num_heads did not improve the final validation loss.
  - The best dropout was seen at 0.2. Although "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" which used of dropout probability = 0.1 in all layers of BERT during fine-tuning (https://arxiv.org/abs/1810.04805) and the documentation for the BERT model class in huggingface (https://huggingface.co/transformers/model_doc/bert.html) passes the dropout probability = 0.1 as the default probability for both self attention and hidden layers.
- The tested numbers of hidden dimensions did not have a significant effect on the final validation loss and perplexity when the best epoch was chosen. However, this dim seemed to have the most impact on runtime.
- It turns out a simple grid search is not likely to result in the most optimal set of hyperparameters due to the computational infeasibility of an exhaustive search in a detailed manner, and there are better methods to explore such as **Bayesian optimization** and **population-based training** according to research.