Matthew Rogers

Homework3 11/8/2024

https://github.com/mrog9/Deep_Learning.git

For the question/answering task, two separate models were used. One model was focused on finding the start index of the answer and the second model was focused on finding the end index of the answer. Both architectures were identical. The difference was how the loss backpropogated through the models. The first "layer" was a base, uncased Bert model which was downloaded using the transformers library. Afterwards, there were five linear layers. The first layer had 768 input nodes and 250 output nodes. The second layer had 100 output nodes. The third layer had 40 output nodes. The fourth layer had 10 output nodes and the fifth layer had 1 output node. All layers had ReLU activation functions except for the last layer.

The SQUAD dataset was used but partially. For the training set, the first 25 documents were used and the first five were used for the testing set. For both sets, the answer was placed in the middle of a 40 word-long window. Then previous windows of same length were used until <PAD> was needed and the same for windows afterwards. Therefore, all the context was used with the answer in its own window. The answer's start and end index were found with respect to all the context windows, not just for one particular window. Only one question and one answer were used for each context paragraph.

The data was moved through the start and end index models using batches. Each batch represented a question and the context for that particular question. For each item in the batch there was the question/context input ids, the attention mask (masked <PAD>), and token type (0s for question and 1 for context). Once the data went through the model, the context was extracted out of the question/context input so that each word of the context had a value associated with it. Then all the windows in the batch were put together in a (1, context_len) shaped tensor. A torch.nn.LogSoftmax() activation function was then used to give the probabilities that a particular index was the start index (or end index). Then, the loss was computed using CrossEntropyLoss for both the inferenced start and end indexes after being compared the true start and end indexes. Also, if the max probability for the end index was greater than the max probability for the start index, the loss for the end model was multiplied by 2.

Both models' weights were initialized using He initialization. Furthermore, a cosine annealing scheduler was used. It started out at a learning rate of 0.000001 and its lowest value was 0.0000001. Only one epoch was used.

To evaluate these models, the testing data was fed through both models in batches. Once the probabilities were calculated for the entire context, the probabilities were reshaped into windows. The max start and end probabilities were found and their sum calculated for each window thereby giving it a score. The window with the highest score was chosen for the answer. Accuracy and f1-score were calculated. The accuracy was 60.2% and the f1-score was 47.47%.

For future experiments, the accuracy and f1-score would need to be increased. Perhaps training on a more diverse dataset with longer training time could increase these scores. Maybe training data could be more preprocessed to make sure the answer is relevant to the question. Trying out different ranges of learning rates and schedulers could also make a difference as well.