Transformer Implementation Report

Tingyu Shi

1 Encoder

1.1 Accuracy

	Epoch 1	Epoch 2	Epoch 3	Epoch 4	Epoch 5	Epoch 6	Epoch 7	Epoch 8
Training Accuracy	44.55 %	51.58 %	58.13 %	63.81 %	75.19 %	78.82 %	79.73 %	91.16 %
Testing Accuracy	33.33 %	49.20 %	48.40 %	55.73 %	65.33 %	70.40 %	68.93 %	80.13 %

Table 1: Training and Testing Accuracy from Epoch 1 to Epoch 8

	Epoch 9	Epoch 10	Epoch 11	Epoch 12	Epoch 13	Epoch 14	Epoch 15 (Final Epoch)
Training Accuracy	91.92 %	95.08 %	95.75 %	97.32 %	97.85 %	96.89 %	97.42 %
Testing Accuracy	82.80 %	82.93 %	83.07 %	85.47 %	86.13 %	85.87 %	85.07 %

Table 2: Training and Testing Accuracy from Epoch 9 to Epoch 15

Table 1 shows the training and testing accuracy after epoch 1 to epoch 8, and table 2 shows the training and testing accuracy after epoch 9 to epoch 15. The accuracy can match what was given in the instruction.
For the training loss, it is 1.0973 after training 1 epoch, and it is 0.0746 after training 15 epochs.

1.2 Sanity Check

The sentence used for generating attention maps is "The Republican nominee, John McCain, has worn the uniform of our country with bravery and distinction, and for that we owe him our gratitude and respect."

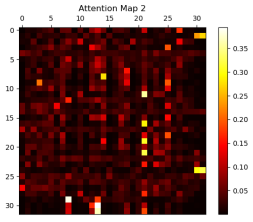

Figure 1: Layer1 Head2

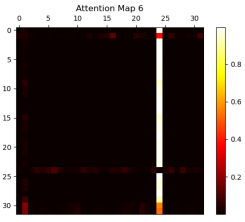

Figure 2: Layer3 Head2

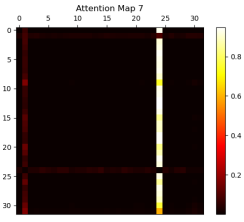

Figure 3: Layer4 Head1

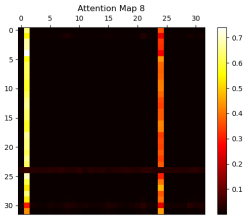

Figure 4: Layer4 Head2

Figure 1 is just a normal attention map and attentions are spread across all tokens, this map shows the correctness of the attention implementaion since all the rows can sum up to 1. Figure 2 and 3

show that all the attention is focused on token "him", which refers to the person described by the sentence. Figure 4 shows that all the attention focused on token "him" and "Republican", which refer to person described by the sentence and an adjective about this person. From the last three attention maps, we can observe that attention mechanism can notice the person that this sentence mainly describes and the adjective corresponding to this person.

2 Decoder

2.1 Perplexity

	After 100 Iters	After 200 Iters	After 300 Iters	After 400 Iters	After 500 Iters
Training Set Perplexity	526.4778	369.1123	273.7272	210.9298	171.5532
Obama Set Perplexity	630.1141	496.7496	424.9127	387.5941	359.7186
HBush Set Perplexity	647.4043	526.4083	460.5224	433.8021	410.7925
WBush Set Perplexity	720.7603	594.7211	532.2336	497.4552	485.7262

Table 3: Decoder Perplexity

Table 3 shows the perplexity on training set and 3 testing sets after every 100 iterations, and after completing all 500 iterations. The perplexity measurements can match what was given in the instruction. The following are some potential reasons for the differences in perplexity for different politicians:

1. Domain Shift: Test dataset with smaller perplexity may have more similar data distribution as the training set compared with test dataset with higher perplexity.

2. OOV Words: The vocabulary presented in the test datasets may differ from the vocabulary seen during training. If a testing dataset contains a large number of out-of-vocabulary words that were not encountered during training, it can result in higher perplexity scores.

3. Linguistic Patterns: Each president may have different language patterns. If the transformer decoder model was trained on a dataset that doesn't adequately capture these individual language patterns, higher perplexity is expected.

2.2 Sanity Check

The sentence used for generating attention maps is "The Republican nominee, John McCain, has worn the uniform of our country with bravery and distinction, and for that we owe him our gratitude and respect." Figure 5 and 6 shows the attention maps of the decoder, which is masked self attention. We can observe that the upper right triangles in two attention maps are all black, which indicates the correct implementation. Also, we can observe that the attention has focused on 9th token, which is "the". We may conclude that common words can gain more attention in the decoder model.

3 Architectural Exploration

In this part, I explored three different kinds of sparse attention. The 3 sparse attention mechanism are Atrous Attention, Local Attention, and Sparse Attention as shown in figure 7.

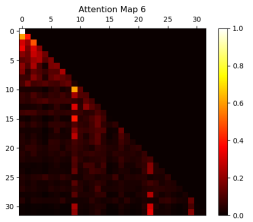

Figure 5: Layer3 Head2

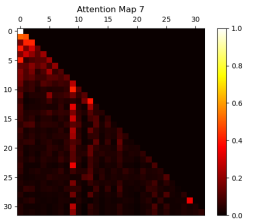

Figure 6: Layer4 Head1

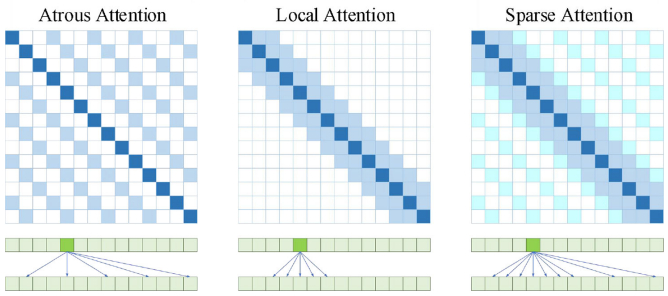

Figure 7: 3 Sparse Attention Mechanism

- Atrous Attention: This attention mechanism forces each token to be associated with only those tokens whose relative distances are $k, 2k, 3k, \cdots$ with $k > 1$ being a predefined hyper-parameter. In our experiment, k is set to 2.
- Local Attention: This attention mechanism constrains each token to be associated only with the preceding and following k tokens as well as itself. In our experiment, k is set to 2.
- Sparse Attention: Sparse attention is simply the combination of atrous attention and local attention.

Table 4 compares the encoder classification accuracy when using different attention mechanisms. We can observe that atrous, local, and sparse attention mechanisms achieved very similar accuracy on both training set and testing set as the full attention mechanism. This experiment demonstrates that atrous, local, and sparse attention mechanisms can achieve similar accuracy as full attention with less computation. NOTE: The actual running time of these 3 experimented attention mechanisms is longer then full attention. This is because for loop has been used in the implementation, so it is slower than fully vectorized implementation as the full attention mechanism.

	Full Attention	Atrous Attention	Local Attention	Sparse Attention
Training Set Accuracy After 15 Epochs	97.42 %	99.09 %	98.52 %	98.52 %
Testing Set Accuracy After 15 Epochs	85.07 %	86.00 %	86.13 %	85.43 %

Table 4: Encoder Accuracy of Different Attention Mechanisms

Figure 8, 9, 10 shows the corresponding attention maps of 3 experimented attention mechanisms.

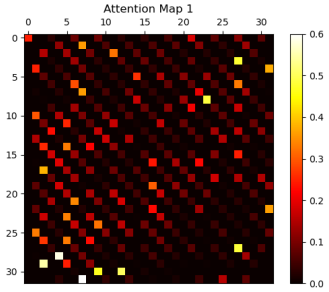

Figure 8: Atrous Attention Map

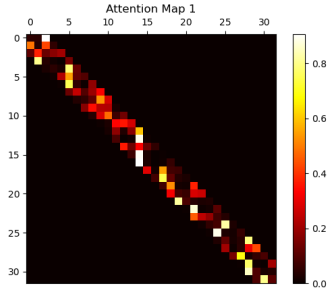

Figure 9: Local Attention Map

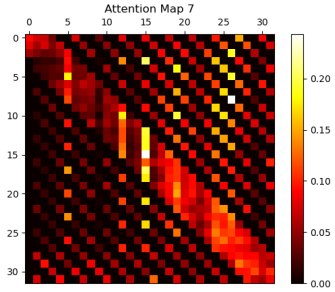

Figure 10: Sparse Attention Map

4 Performance Improvement

For the performance improvement, I chose to improve the performance of decoder. I fine-tuned the following hyperparameters. The other hyperparameters were kept as the default.

1. Learning Rate = $\{5 \times 10^{-4}, 1 \times 10^{-3}, 2 \times 10^{-3}\}$
2. Token Embedding Dimension(n_embds) = $\{32, 64, 128\}$
3. Number of Attention Heads(n_head) = $\{2, 4, 8\}$
4. Dropout Probabilities = $\{0.2, 0.4, 0.6, 0.8\}$
5. Include Learning Rate Scheduler = $\{True, False\}$. The learning rate scheduler here is a step scheduler, it will halve the learning rate every 200 iterations.

There are in total 216 different combinations of the above hyperparameters. For each combination, I tested the model performace on training set, Obama testing set, HBush testing set, and WBush testing set. Table 5 shows the best hyperparameter combination for the training set and 3 testing sets.

	LR	n_embds	n_head	Dropout Probabilities	Include LR Scheduler	Best Perplexity
Training Set	0.002	128	4	0.2	$False$	56.4394
Obama Testing Set	0.001	128	4	0.2	$True$	310.0157
HBush Testing Set	0.001	128	8	0.2	$True$	349.1012
WBush Testing Set	0.001	128	2	0.2	$True$	415.3683

Table 5: Best Hyperparameter Combination for Different Datasets

From table 5, we can gain the following insights:

1. Increasing token embedding dimension can lead to better model performance. In this experiment, all four best hyperparameter combinations use the largest token embedding dimension.
2. Including LR schedulers can help to achieve better testing datasets perplexity because they can help to avoid the overfitting issues.
3. In this experiment, 0.2 dropout probability is the best choice for training set and 3 testing sets.
4. We cannot directly observe any associations between number of attention heads and the model performance.