Advanced Natural Language Processing

Theory Questions

What is the purpose of self-attention, and how does it facilitate capturing dependencies in sequences?

> - Self-attention, is a mechanism that allows models to weigh the significance of different words in a sentence relative to each other. Unlike traditional recurrent neural networks (RNNs) and convolutional neural networks (CNNs), which process sequences sequentially or with fixed-size receptive fields, self-attention enables parallel processing of tokens within a sequence. This parallelism not only facilitates efficient computation but also enables the model to capture long-range dependencies.
> - The mechanism of self-attention can be visualized as a process of information exchange among tokens within a sequence. Given an input sequence of tokens, typically represented as embeddings, self-attention computes three key components: Query, Key, and Value. These components are linear projections of the input embeddings, which are then used to calculate attention scores. The attention scores determine the importance of each token in relation to others in the sequence. The weighted sum of values, where the weights are determined by the attention scores, yields the attended representation of each token.
> - Hence self-attention enhances a model's ability to capture dependencies and relationship between the words in a sequence by enabling parallel processing allowing efficient and better understanding of the concepts.

Why do transformers use positional encodings in addition to word embeddings? Explain how positional encodings are incorporated into the transformer architecture. Briefly describe recent advances in various types of positional encodings used for transformers and how they differ from traditional sinusoidal positional encodings.

> - The self-attention mechanism treats words equally, lacking inherent knowledge of their order. To address this, positional encodings are added to the word embeddings in transformers.
> - Positional encodings are the solution to convey the position of words within a sequence to the Transformer model. Instead of relying solely on the sequential order of the words, traditionally these encodings are generated using a combination of sine and cosine functions.

> - The positional encoding for a word at position pos and dimension i can be computed as follows (pos represents the word’s position in the sequence, i corresponds to the dimension within the embedding, and d_model is the dimensionality of the model’s embeddings) :

$$\mathrm{PE}_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{\mathrm{model}}}}\right)$$

$$\mathrm{PE}_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{\mathrm{model}}}}\right)$$

> - Recently various other types of Positonal Encodings have been introduced -
>
> 1. Learned Positional Encodings - Unlike sinusoidal encodings, which are fixed and computed using trigonometric functions, learned positional encodings are parameters that are optimized during training. It allows the model to adapt the postional information according to the data.
> 2. Relative Positional Encodings - These encodings focus on relative distances between tokens in a sequence rather than absolute positions as in case of sinusoidal encodings.

Hyperparameter Tuning and Analysis

Number of Layers in the Encoder Decoder Stack

- The number of layers in the encoder and decoder determines the complexity of the Transformer model. Each layer in the encoder/decoder is made up of multi-head self-attention mechanisms and feed-forward neural networks.
- More layers increase the model's ability to learn complex patterns but also increase computational cost and the risk of overfitting on smaller datasets whereas fewer layers reduce model complexity but may lead to underfitting, where the model cannot capture all necessary patterns in the data.
- Since our dataset is small we use only 6 Layers or Blocks

Number of Attention Heads

- The number of attention heads determines how many parallel attention mechanisms are used in the multi-head self-attention layers. Each attention head learns to focus on different parts of the input sequence and learn different types of relationship between the words.
- Increasing the number of heads gives the model more "views" of the data, improving its ability to model complex sequences. However, it also increases memory consumption and computation time. If set too high, it could lead to overfitting or model instability.

Embedding Dimensions

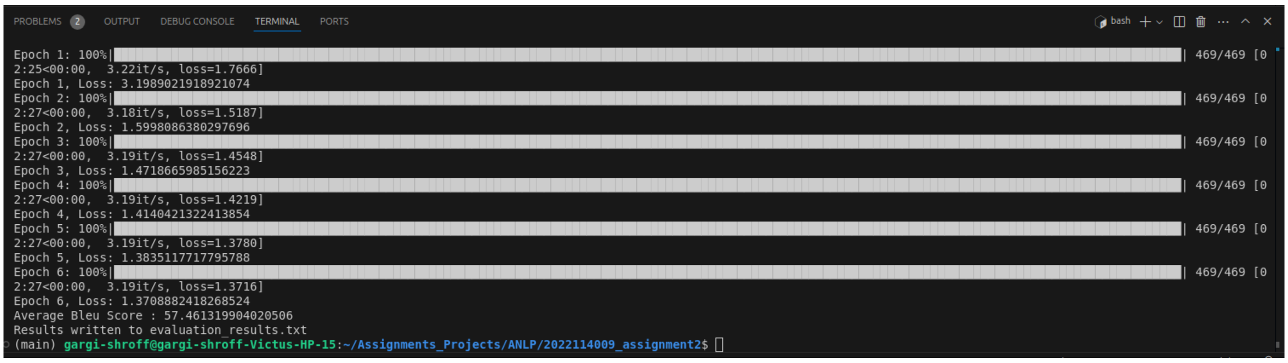
- Larger embeddings can capture more detailed information about word relationships and semantics, which is helpful for larger vocabularies and more complex tasks. However, they also increase model size, making the model slower to train and more prone to overfitting. For this task we use 512 as Embedding Dimension of the Model and 2048 as the dimension of the feedforward layer.

Dropout

- Dropout is a regularization technique used to prevent overfitting. It randomly sets a fraction of the neurons to zero during training, making the model less sensitive to specific neurons.
- Higher dropout values (e.g., 0.3–0.5) introduce more regularization, helping the model generalize better on unseen data. Lower dropout values (e.g., 0.1 or no dropout) can lead to faster training and more memorization of training data, which might improve performance on the training set but cause overfitting.

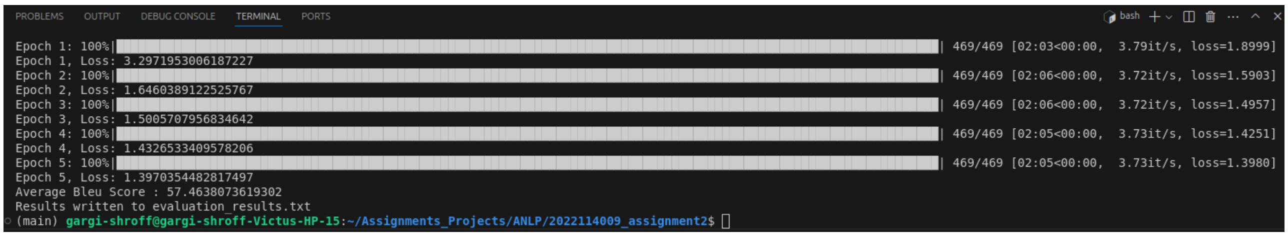
Results of Hyperparameter Tuning

1. MODEL_DIM = 512, FFD = 2048, NUM_HEADS = 8, NUM_LAYERS = 6, DROPOUT = 0.1



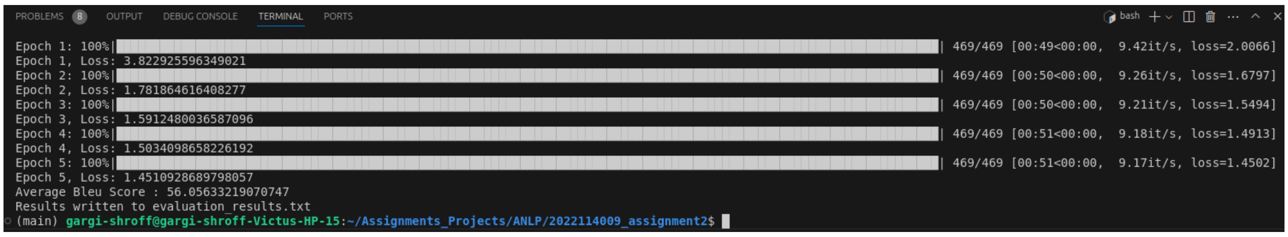
```
Epoch 1: 100%|████████████████████████████████████████████████████████████| 469/469 [0
2:25<00:00,  3.22it/s, loss=1.7666]
Epoch 1, Loss: 3.1989021918921074
Epoch 2: 100%|████████████████████████████████████████████████████████████| 469/469 [0
2:27<00:00,  3.18it/s, loss=1.5187]
Epoch 2, Loss: 1.5998086380297696
Epoch 3: 100%|████████████████████████████████████████████████████████████| 469/469 [0
2:27<00:00,  3.19it/s, loss=1.4548]
Epoch 3, Loss: 1.4718665985156223
Epoch 4: 100%|████████████████████████████████████████████████████████████| 469/469 [0
2:27<00:00,  3.19it/s, loss=1.4219]
Epoch 4, Loss: 1.4140421322413854
Epoch 5: 100%|████████████████████████████████████████████████████████████| 469/469 [0
2:27<00:00,  3.19it/s, loss=1.3780]
Epoch 5, Loss: 1.3835117717795788
Epoch 6: 100%|████████████████████████████████████████████████████████████| 469/469 [0
2:27<00:00,  3.19it/s, loss=1.3716]
Epoch 6, Loss: 1.3708882418268524
Average Bleu Score : 57.461319904020506
Results written to evaluation_results.txt
(main) gargi-shroff@gargi-shroff-Victus-HP-15:~/Assignments_Projects/ANLP/2022114009_assignment2$
```

2. MODEL_DIM = 512, FFD = 1024, NUM_HEADS = 8, NUM_LAYERS = 6, DROPOUT = 0.2



```
Epoch 1: 100%|██████████████████████████████████████████████| 469/469 [02:03<00:00,  3.79it/s, loss=1.8999]
Epoch 1, Loss: 3.2971953006187227
Epoch 2: 100%|██████████████████████████████████████████████| 469/469 [02:06<00:00,  3.72it/s, loss=1.5903]
Epoch 2, Loss: 1.6460389122525767
Epoch 3: 100%|██████████████████████████████████████████████| 469/469 [02:06<00:00,  3.72it/s, loss=1.4957]
Epoch 3, Loss: 1.5005707956834642
Epoch 4: 100%|██████████████████████████████████████████████| 469/469 [02:05<00:00,  3.73it/s, loss=1.4251]
Epoch 4, Loss: 1.4326533409578206
Epoch 5: 100%|██████████████████████████████████████████████| 469/469 [02:05<00:00,  3.73it/s, loss=1.3980]
Epoch 5, Loss: 1.3970354482817497
Average Bleu Score : 57.4638073619302
Results written to evaluation_results.txt
(main) gargi-shroff@gargi-shroff-Victus-HP-15:~/Assignments_Projects/ANLP/2022114009_assignment2$
```

3. MODEL_DIM = 256, FFD = 1024, NUM_HEADS = 4, NUM_LAYERS = 4, DROPOUT = 0.2



```
Epoch 1: 100%|██████████████████████████████████████████████| 469/469 [00:49<00:00,  9.42it/s, loss=2.0066]
Epoch 1, Loss: 3.822925596349021
Epoch 2: 100%|██████████████████████████████████████████████| 469/469 [00:50<00:00,  9.26it/s, loss=1.6797]
Epoch 2, Loss: 1.781864616408277
Epoch 3: 100%|██████████████████████████████████████████████| 469/469 [00:50<00:00,  9.21it/s, loss=1.5494]
Epoch 3, Loss: 1.5912480036587096
Epoch 4: 100%|██████████████████████████████████████████████| 469/469 [00:51<00:00,  9.18it/s, loss=1.4913]
Epoch 4, Loss: 1.5034098658226192
Epoch 5: 100%|██████████████████████████████████████████████| 469/469 [00:51<00:00,  9.17it/s, loss=1.4502]
Epoch 5, Loss: 1.4510928689798057
Average Bleu Score : 56.05633219070747
Results written to evaluation_results.txt
(main) gargi-shroff@gargi-shroff-Victus-HP-15:~/Assignments_Projects/ANLP/2022114009_assignment2$
```

Best Bleu Score of 0.57 is obtained for MODEL_DIM = 512, FFD = 1024, NUM_HEADS = 8, NUM_LAYERS = 6 AND DROPOUT = 0.2