Practical Report IFT6135-A2022
Assignment 2

Hamed Nazim MAMACHE

November 2022

1 Problem 2

5. How many learnable parameters does your module have, as a function of num heads and head size?

Answer

In the Transformer's MultiHeadAttention, the input vector is transformed in multiple heads, then applied the self-attention operation, then all are concatenated, and then a fully connected dense forward layer is applied. Let's see how many learnable parameters there are.

The input vector of dimension `head_size` gets multiplied by three matrices W_Q, W_K, W_V, `num_heads` times. That give $3 \times$ `num_heads` vectors (q, k, v). So dimension of each of these matrices is $\texttt{head\_size} \times \frac{\texttt{head\_size}}{\texttt{num\_heads}}$ and we have $3 \times$ `num_heads` such matrices.
Including the bias for eah of Q, K, V matrices, we then have :

$$\texttt{head\_size} \times \frac{\texttt{head\_size}}{\texttt{num\_heads}} \times 3 \times \texttt{num\_heads} + \texttt{head\_size} \times 3$$

These are then concatenated, and passed through the dense layer W_O which would have dimension $\texttt{head\_size} \times \texttt{head\_size} + \texttt{head\_size}$ (with bias b_O).

So, the final number of learnable parameters is equal to:

$$(3 \times \texttt{num\_heads} \times (\texttt{head\_size} \times \frac{\texttt{head\_size}}{\texttt{num\_heads}}) + 3 \times \texttt{head\_size}) + (\texttt{head\_size} \times \texttt{head\_size} + \texttt{head\_size})$$

2 Problem 3

1. You are asked to run 8 experiments with different architectures, optimizers, and decoding strategies. These parameter settings are given to you as separate cells in the notebook. For each of these 8 experiments, plot learning curves (train and validation) of the loss and perplexity over both epochs and time. Figures should have labeled axes and a legend and an explanatory caption.

Answer

For each of these 8 experiments, we can here observe learning curves (train and validation) of the loss and perplexity over both epochs and time.

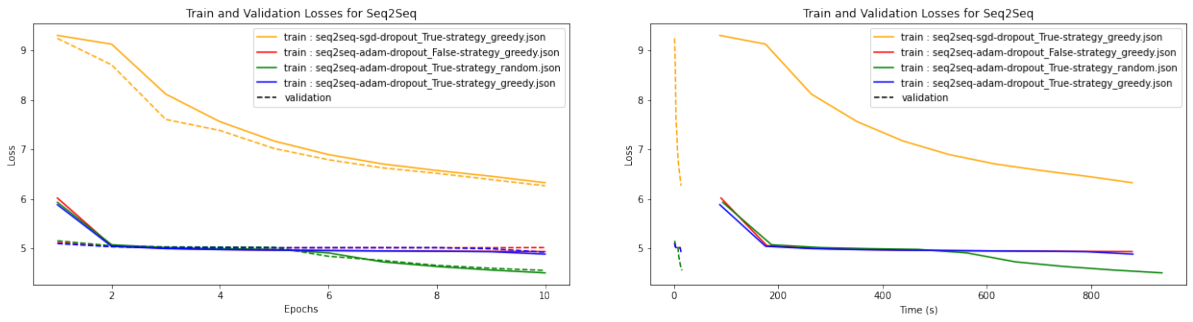


Figure 1: GRU-based model : Evolution of train and validation loss over epochs (left) and time (right)

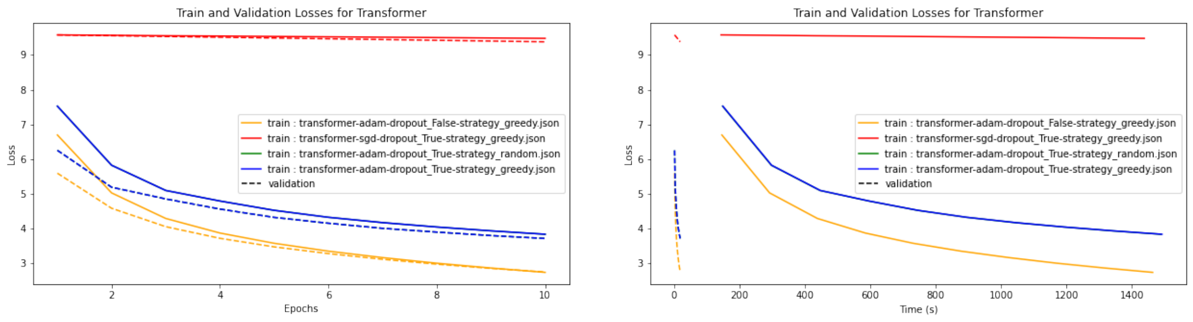


Figure 2: Transformer : Evolution of train and validation loss over epochs (left) and time (right)

Note : Less overloaded graphs can be found in the appendix.

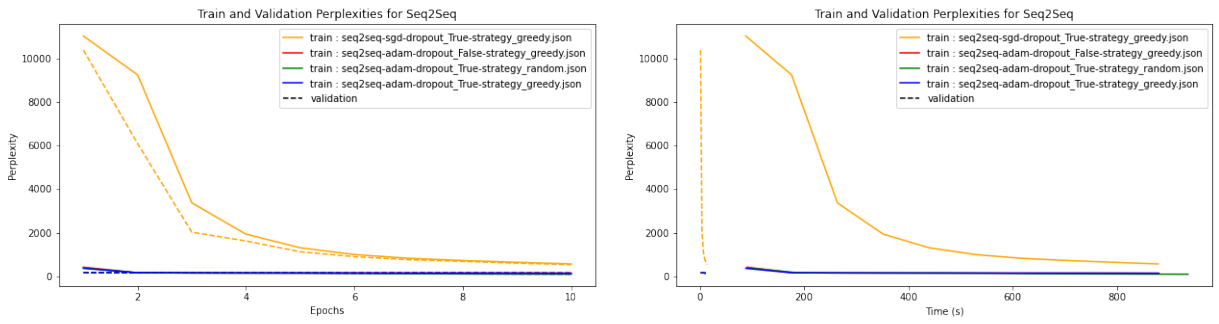


Figure 3: GRU-based model : Evolution of train and validation perplexity over epochs (left) and time (right)

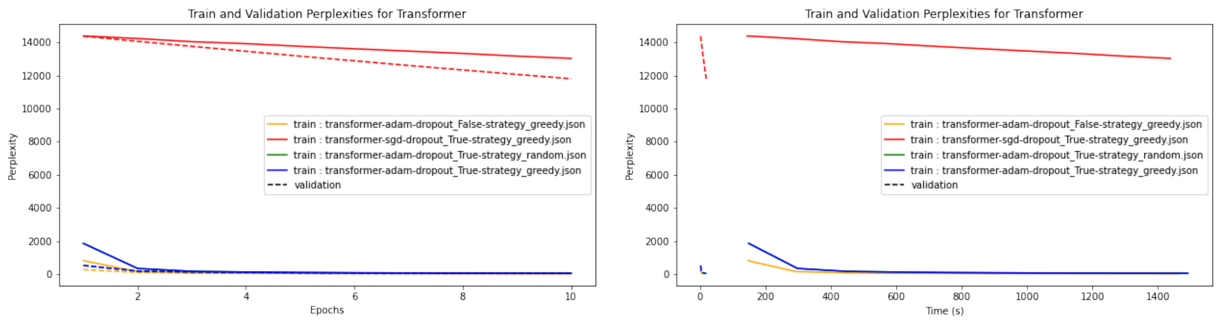


Figure 4: Transformer : Evolution of train and validation perplexity over epochs (left) and time (right)

2. Make a table of results summarizing the train, validation and test performance for each experiment, indicating the architecture, optimizer, dropout and decoding strategy. Sort by architecture, then use dropout, then optimizer and finally by decoding strategy. Bold the best result for each architecture. You will report loss and perplexity (best across epochs) and the final test BLEU score.

   Answer

   Table of results, sorted by architecture, then use_dropout, then optimizer and finally by decoding strategy. Times, percentage of used memory, loss, perplexity (best across epochs) and the final test BLEU score are reported.
   The best result for each architecture are in bold.

Architecture	Dropout	Optimizer	Decoding strategy	Time (seconds)	Memory used (%)	Best Loss	Best Perplexity	BLEU score
GRU-based mode	False	Adam	greedy	train : 879.12 validation : 13.73 test : 1.36	3242616422.4 (20.47 %)	train : 4.94 validation : 5.02 test : 5.00	train : 140.09 validation : 151.02 test : 149.12	test bleu1 : 24.05 test bleu2 : 14.34
GRU-based mode	True	Adam	greedy	train : 880.58 validation : 13.77 test : 1.36	3474980864 (21.93 %)	train : 4.89 validation : 4.93 test : 4.90	train : 132.99 validation : 138.39 test : 135.44	test bleu1 : 27.18 test bleu2 : 16.11
GRU-based mode	**True**	**Adam**	**random sampling with temperature**	**train : 936.29** **validation : 15.67** **test : 1.56**	**3474980864** **(21.93%)**	**train : 4.51** **validation : 4.56** **test : 4.53**	**train : 91.07** **validation : 95.68** **test : 92.99**	**test bleu1 : 33.80** **test bleu2 : 16.87**
GRU-based mode	True	SGD	greedy	train : 879.21 validation : 13.78 test : 1.44	3473303142.4 (21.92 %)	train : 6.33 validation : 6.27 test : 6.29	train : 562.82 validation : 528.98 test : 538.51	test bleu1 : 3.54 test bleu2 : 0.0
Transformer	**False**	**Adam**	**greedy**	**train : 1463.08** **validation : 19.06** **test : 1.86**	**6880755712.0** **(43.43%)**	**train : 2.74** **validation : 2.75** **test : 2.69**	**train : 15.55** **validation : 15.71** **test : 14.78**	**test bleu1 : 45.30** **test bleu2 : 59.46**
Transformer	True	Adam	greedy	train : 1491.60 validation : 19.11 test : 1.87	7677673472 (48.46 %)	train : 3.84 validation : 3.72 test : 3.70	train : 46.57 validation : 41.26 test : 40.60	test bleu1 : 23.06 test bleu2 : 36.62
Transformer	True	Adam	random sampling with temperature	train : 1488.68 validation : 19.05 test : 1.87	7709759897.6 (48.66 %)	train : 3.84 validation : 3.72 test : 3.70	train : 46.57 validation : 41.26 test : 40.60	test bleu1 : 22.06 test bleu2 : 37.82
Transformer	True	SGD	greedy	train : 1437.55 validation : 19.12 test : 1.87	7642860748.8 (48.24 %)	train : 9.48 validation : 9.38 test : 9.37	train : 13035.19 validation : 11803.99 test : 11695.51	test bleu1 : 0.0 test bleu2 : 0.03

Table 1: Table for all the experiments; Time (s), Average Memory Used (%), Best Training and Validation Loss and Perplexity have been given (10 epochs have been run). The best results for each architecture are in bold.

3. Which model and configuration would you use if you were most concerned with time? With generalization performance? What was the impact of Dropout?

   Answer

   If I was most concerned with time, the best model would be the GRU-based model with drop-out set on False, Adam Optimizer and greedy decoding strategy.
   However, it is known that the Transformer can be parallelized, and can therefore be faster than during our experiment.
   If I was most concerned with generalization performance, the best model would be the Transformer with drop-out set on False, Adam Optimizer and greedy decoding strategy.
   Dropout is a technique that prevents overfitting in artificial neural networks by randomly dropping units during training. As a result, the trained model works as an ensemble model consisting of multiple neural networks. Dropout has the effect of making the training process noisy, forcing nodes within a layer to probabilistically take on more or less responsibility for the inputs.
   However, in our case, the best results are obtained when the dropout is not used (with Transformer model), maybe because overfitting is not a concern here, after 10 epochs.

4. Compare the GRU and Transformer based on the experiment configuration: Adam optimizer, `use_dropout=False`, greedy decoding. Which model did you think performed better? Why?

   Answer

   Generally, with the same configurations, the transformer always shows the better results than the GRU-based model (this is not the case only when considering the SGD optimizer).
   In addition, we note that in the case of the GRU-based model, the BLEU 1 score is higher than the BLEU 2 score, unlike the Transformer where the BLEU 2 score is the highest.
   I expect that the Transformer performed better because the Transformer use non-sequential processing (sentences are processed as a whole, rather than word by word) and, more particularly, it is able to pay attention to every previous single aspect of the input in a direct way, unlike the GRU-based model.
   In fact, in GRU-based model, if we consider a sentence with n words for example, the last word receives only indirect information about the first word and is thus not able to 'look' at this word directly.

5. You have trained a transformer with a few different configurations in this experiment. Given the recent high profile transformer-based language models, are the results as you expected? Speculate as to why or why not.

   Answer

   I initially expected the transformer to perform better than all the models considering it uses attention to all the previous words in the sentence. This experience made it possible to validate my expectations: the transformer generally presents the best results.
   Moreover, it is worth pointing out that in the case of the GRU-based model, the BLEU score

1 is always higher than the BLEU score 2, unlike the Transformer, which has better BLEU score 2.

6. For each of the experiment configurations above, measure the average steady-state GPU memory usage (`nvidia-smi` is your friend!). Comment about the GPU memory footprints of each model, discussing reasons behind increased or decreased memory consumption where applicable.

Answer

The average steady-state GPU memory usage are reported in Table 1. For each of the experiment configurations above, we can here observe the evolution of the memory used, in percentage.
As the average values in the table and the graphs show, the transformer uses much more memory than the GRU-based model (more than double!). This seems rather coherent to me since the transformer seeks to preserve long-term dependencies in long text sequences; and it is memory intensive because self attention is computation-hungry.
In addition, we notice that the memory used on average is lower when we do not use drop out. As said before, when using the drop out, the trained model works as an ensemble model consisting of multiple neural networks. Generating multiple neural network every time using dropout could be the reason for the extra memory usage.

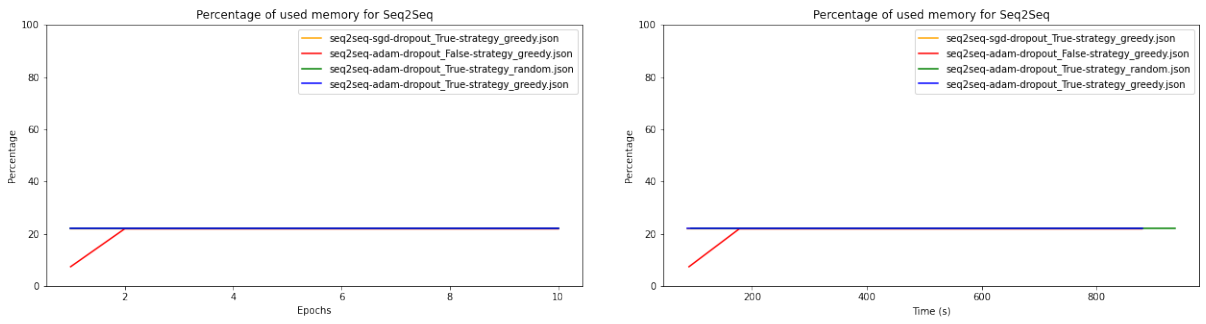
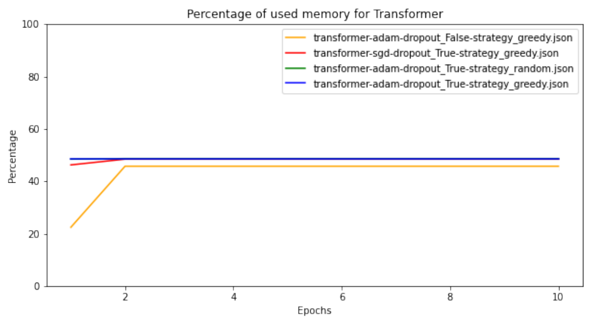


Figure 5: GRU-based model : Evolution of percentage of used memory over epochs (left) and time (right)



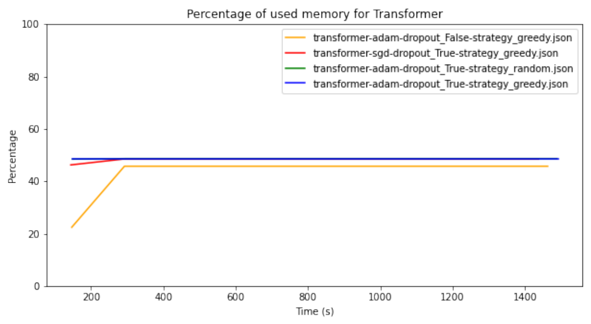


Figure 6: Transformer : Evolution of percentage of used memory over epochs (left) and time (right)

7. Comment on the overfitting behavior of the various models you trained. Did a particular class of models overfit more easily than the others? Can you make an informed guess of the various steps a practitioner can take to prevent overfitting in this case?

Answer

I initially expected that models not using dropout would present more pronounced overfitting than models using dropout.
As can be seen better in the figures in the appendix, almost no model has overfitting. Indeed, we can observe a beginning of overfitting for the GRU-based model, Adam optimize, use_dropout False, and greedy strategy.
Moreover, we could expect not to observe any overfitting, since we train ”only” on 10 epochs.
Solutions to prevent overfitting would be the use of dropout, the use of regularization, or increase the size of the training dataset (using data augmentation for example).

3 Appendix

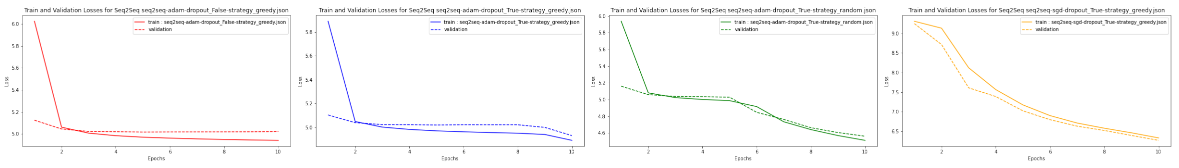


Figure 7: GRU-based model : train and validation loss curves over epochs

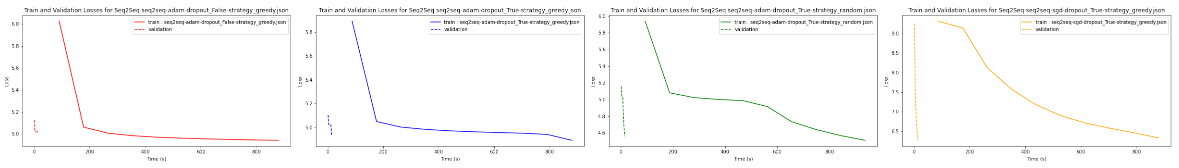


Figure 8: GRU-based model : train and validation loss curves over time

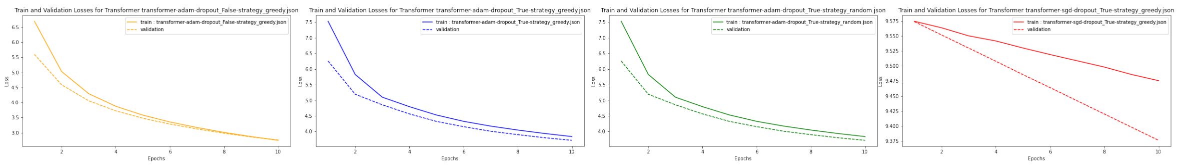


Figure 9: Transformer : train and validation loss curves over epochs

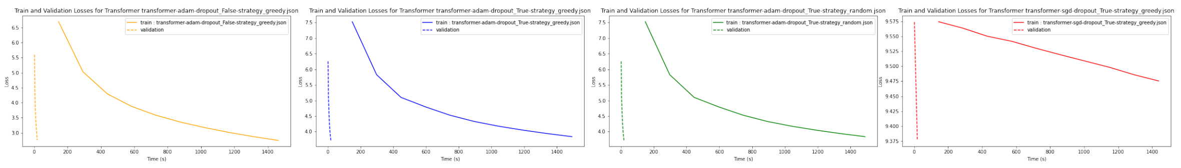


Figure 10: Transformer : train and validation loss curves over time

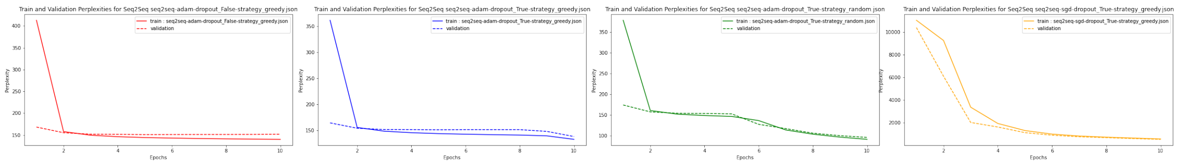


Figure 11: GRU-based model : train and validation perplexity curves over epochs

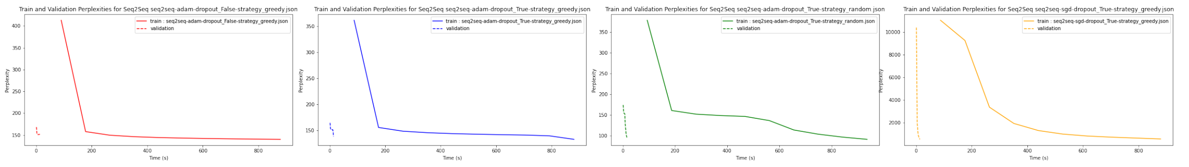


Figure 12: GRU-based model : train and validation perplexity curves over time

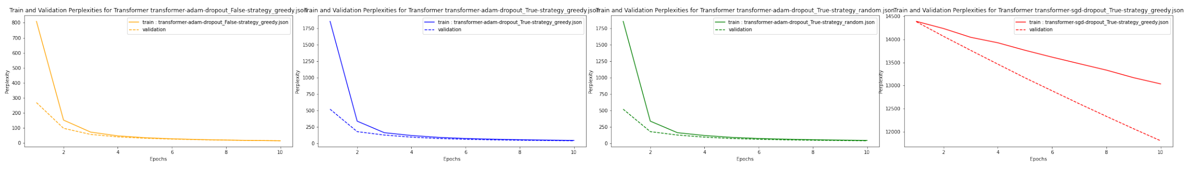


Figure 13: Transformer : train and validation perplexity curves over epochs

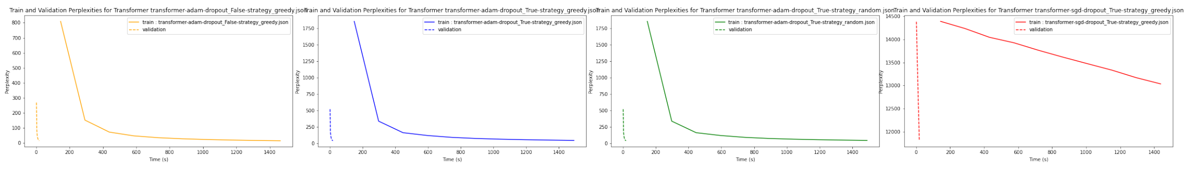


Figure 14: Transformer : train and validation perplexity curves over time