Seq2Seq Results

- Put your results from training before and after hyperparameter tuning here.

	Default (**RNN**)	1st tuning (**LSTM**)	2nd tuning (lr=1e-1, **1e-2**, **le-3**, le-4)	3rd tuning (en_emb_size=**16**, **32**)	4th tuning (de_emb_size=16, **32**)
Training Loss	4.5983	4.1681	3.7720	3.7295	3.7925
Validation Loss	4.5225	4.0193	3.7264	3.6761	3.7541
Training Perplexity	99.3115	64.5932	43.4667	41.6581	44.3689
Validation Perplexity	92.0621	55.6596	41.5294	39.4924	42.6940
	5th tuning (ende_hidsize=32,**64**)	6th tuning (ende_hidsize=**64**,**128**)	7th tuning (ende_dropout=0.1, **0.2**)	8th tuning (ende_dropout=**0.2**, 0.3)	9th tuning (epoch=**10**, 20)
Training Loss	4.4030	**3.6454**	3.8759	3.6602	3.3600
Validation Loss	4.3361	**3.6353**	3.8799	3.6690	3.7007
Training Perplexity	53.2475	**38.2990**	48.2239	38.8682	28.7887
Validation Perplexity	50.1978	**37.9122**	48.4215	39.2130	40.4751

*Black bold text is the default values

*Red bold text is the hyperparameter chosen

Seq2Seq Results (continued)

- Explain what you did here, you can use another slide if needed

- After running RNN with default hyperparameters, I got the result of 4.5225 for validation loss and 92.0621 for validation perplexity. First tuning was done by changing RNN to LSTM. There was a significant decrease in the perplexity which is 55.6596 and validation loss of 4.0193. The first hyperparameter that I tuned was the learning rate. Default was 1e-3 and I tried 1e-1, 1e-2, and 1e-4. Using higher learning rate enables the training to have bigger steps. Therefore, it can learn faster but there is a possibilities that the result will fall into local maxima. By using higher parameter 1e-2, I was able to get higher validation perplexity of 41.5294. Moreover, decreasing encoder embedding size gave better result which had 39.4924 for validation perplexity. My hypothesis was that we will have better result if we both decrease encoder and decoder embedding sizes, but the result was different. Decreasing the size of decoder embedding didn't improve the result. Hidden size of encoder and decoder also affected the result. Increasing the hidden size to 128 (default 64) made the validation perplexity of 37.9122. This is a very possible result since number of features of the hidden state increased and therefore, there are more features to be trained. Furthermore, Changing dropout rate didn't result in improving the overall perplexity. The last hyperparameter tuned is the epoch. I tried to increase the epoch from 10 to 20. The training perplexity decreased a lot with the value 28.7887 but the validation perplexity increased to 40.4751. Training perplexity had better result but when looking at the data, validation perplexity could not follow the training perplexity which results overfitting. As a result, the best losses and perplexities are as follows. Training perplexity: 38.2990, Validation perplexity: 37.9122, Training loss: 3.6454, Validation loss: 3.6353. Compared to the default parameters, there was a significant decrease of perplexities and losses.

Transformer Results

- Put your results from training before and after hyperparameter tuning here

	Default (**Epoch=10**, **lr=1e-3**)	1st tuning (**Epoch=10**, **lr=1e-2**)	2nd tuning (**Epoch=10**, lr=1e-4)	3rd tuning (amsgrad=True)	4th tuning (eps=1e-7)
Training Loss	2.2879	3.0573	3.6439	2.2918	2.2824
Validation Loss	3.0021	3.4174	3.5107	3.0013	3.0096
Training Perplexity	9.8542	21.2710	38.2424	9.8931	9.8000
Validation Perplexity	20.1277	30.4909	33.4719	20.1111	20.2786
	5th tuning (**eps=1e-10**)	6th tuning (beta=(0.5,0.999))	7th tuning (beta=(0.9,0.9999))	8th tuning (epoch=20)	9th tuning (epoch=30)
Training Loss	**2.2851**	2.3313	2.2670	1.5558	1.1385
Validation Loss	**2.9994**	2.9607	3.0040	3.5009	4.2196
Training Perplexity	**9.8265**	10.2913	9.6506	4.7388	3.1219
Validation Perplexity	**19.4706**	19.3122	20.1660	33.1464	68.0047

*Black bold text is the default values

*Red bold text is the hyperparameter chosen

Transformer Results (continued)

- Explain what you did here as well, you can use another slide if needed

  - After running the transformer with the default parameters, I got training loss of 2.2879, validation loss of 3.0021, training perplexity of 9.8542 and validation perplexity of 20.1277. First hyperparameter to modify was the learning rate. 1e-4, 1e-3 and 1e-1 was tested but the best result was 1e-2 which was the default. This means the learning rate is good enough and there is no need for a smaller or bigger steps. Second tuning was done by changing the amsgrad parameter from false to true. Amsgrad is one of the stochastic optimization method used to fix the convergence issue. However, using this optimization method didn't change the result that much. The result was almost the same with the previous results. Third parameter tuned was the eps. Eps is the epsilon for numerical stability. I tried to increase and decrease this stability. I could conclude that using smaller epsilon produce better stability and therefore better perplexities. Using epsilon of 1e-10 had validation perplexity of 19.4706. Moreover, beta values were tuned. Tried different sets of beta values but didn't have impacts on the results. Last parameter that is tuned was the epoch. I tried epoch of 20 and 30. As the number of epoch increased, training perplexity decreased but validation perplexity increased. I was able to observe that as number of epoch increase, there is a high change of overfitting. Validation perplexity could not follow the rate of training perplexity. In other words, it worked well only with the training data. If new data comes in, it has performance issues since it is only doing well with the training data.