Assignment 3 (NLP) Writeup

DO NOT TAG

Name: Trung Pham
GT Email: tpham328@gatech.edu

Seq2Seq Results – Default configuration
Values are for last epoch

RNN
Training Loss: 4.608
Training Perplexity: 100.314
Validation Loss: 4.700
Validation Perplexity: 109.931

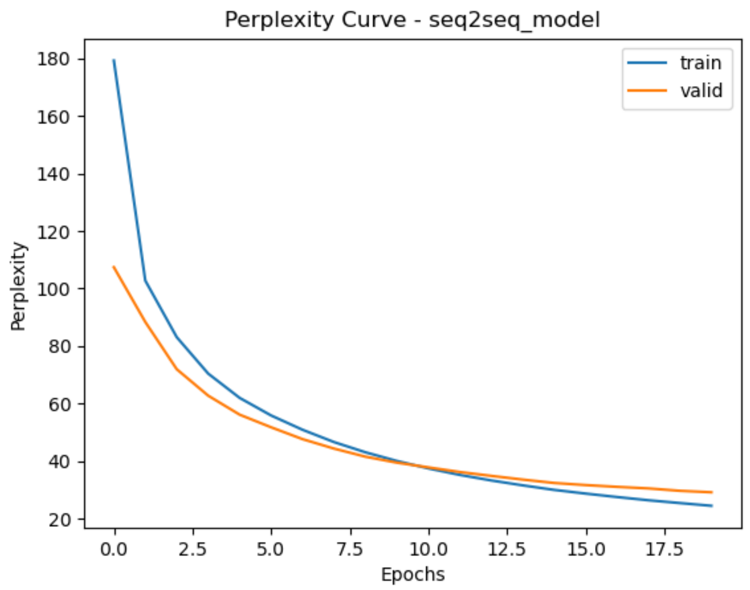
LSTM
Training Loss: 3.200
Training Perplexity: 24.523
Validation Loss: 3.375
Validation Perplexity: 29.215

RNN-with-Attention
Training Loss: 3.300
Training Perplexity: 27.063
Validation Loss: 3.441
Validation Perplexity: 31.218

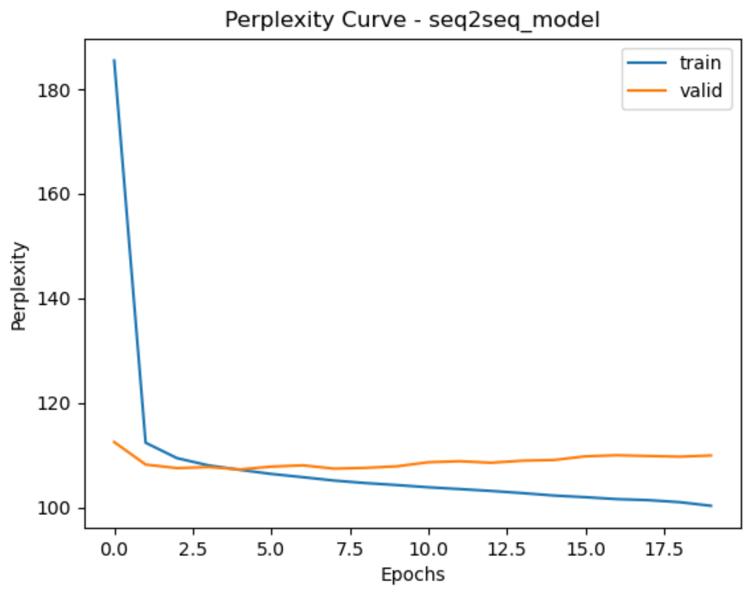
LSTM-with-Attention
Training Loss: 3.099
Training Perplexity: 22.174
Validation Loss: 3.304
Validation Perplexity: 27.225

Seq2Seq Explanation (RNN vs LSTM)

Compare your RNN result to your LSTM result and explain why they differ.



LSTM



RNN

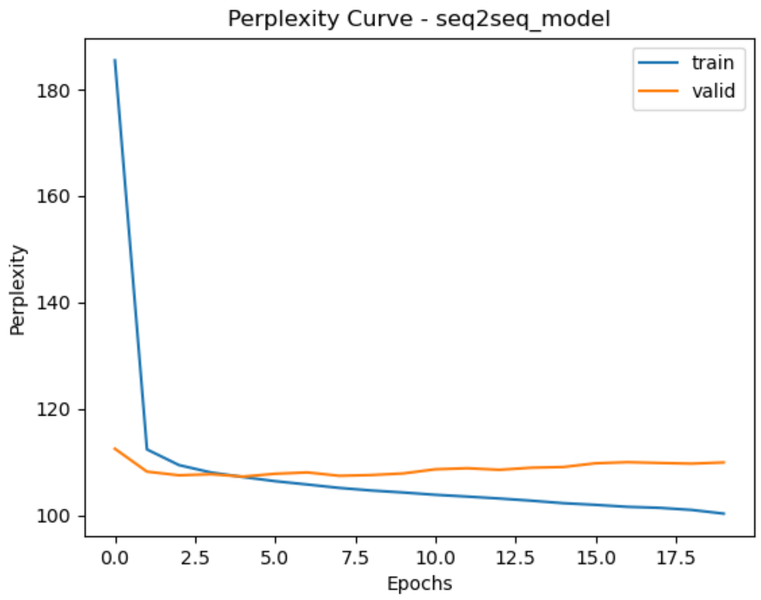
Seq2Seq Explanation (RNN vs LSTM)

Compare your RNN result to your LSTM result and explain why they differ.

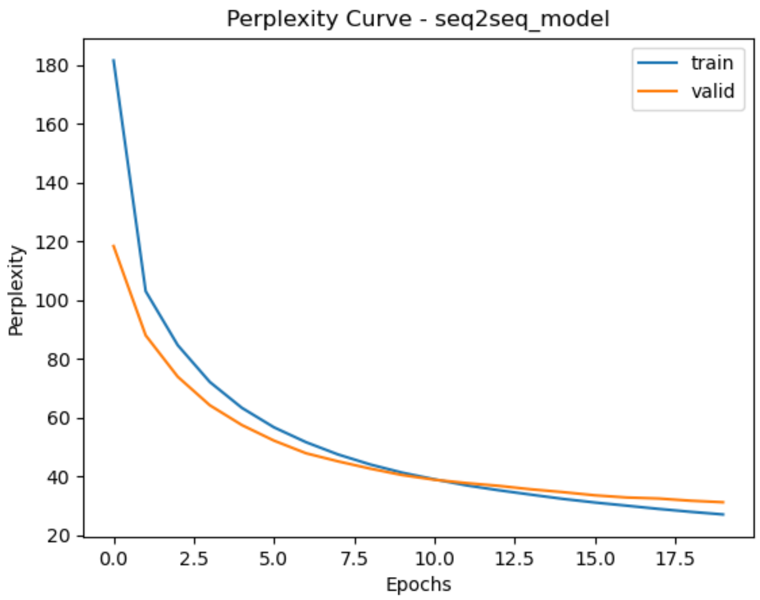
In RNN, the training perplexity decreases steadily as epoch increase but validation score start deceasing slowly and diverge, this indicates overfitting. This is typical because we expect vanishing gradients in RNN model.
LSTM handle vanishing gradients better than RNN, this indicate by the validation perplexity go closer and more resemble of the training line. This solve the overfitting issue in RNN case.
RNN perplexity is higher , while LSTM validation perplexity is smaller, indicating LSTM have better performance.
In conclusion, LSTM outperform RNNs , better result, less overfitting and better generalization because LSTM handle long-range dependencies better.

Seq2Seq Explanation (RNN vs RNN-with-Attention)

Compare your RNN result to your RNN-with-Attention result and explain why they differ.



RNN without attention



RNN with attention

Seq2Seq Explanation (RNN vs RNN-with-Attention)

Compare your RNN result to your RNN-with-Attention result and explain why they differ.

As being mentioned earlier, the RNNs model without attention is struggling due to vanishing gradient problem. It is overfitting (2 lines separate and diverge), the perplexity result is high, and model struggle with generalization.

RNNs with attention perform much better than RNNs without. Validation curve goes in line with training curve, and still decreasing. This indicate that the model generalize better, and is less overfitting.
The perplexity is also significantly lower, so better performance.
The attention algorithm helps the model focus on relevant parts of the input and alleviate the long-term memory problem that vanilla RNNs has.

Seq2Seq Results – Best model
Values are for last epoch

Your best model after hyper-parameter tuning

Best model

Training Loss: 2.469

Training Perplexity: 11.809

Validation Loss: 3.213

Validation Perplexity: 24.854

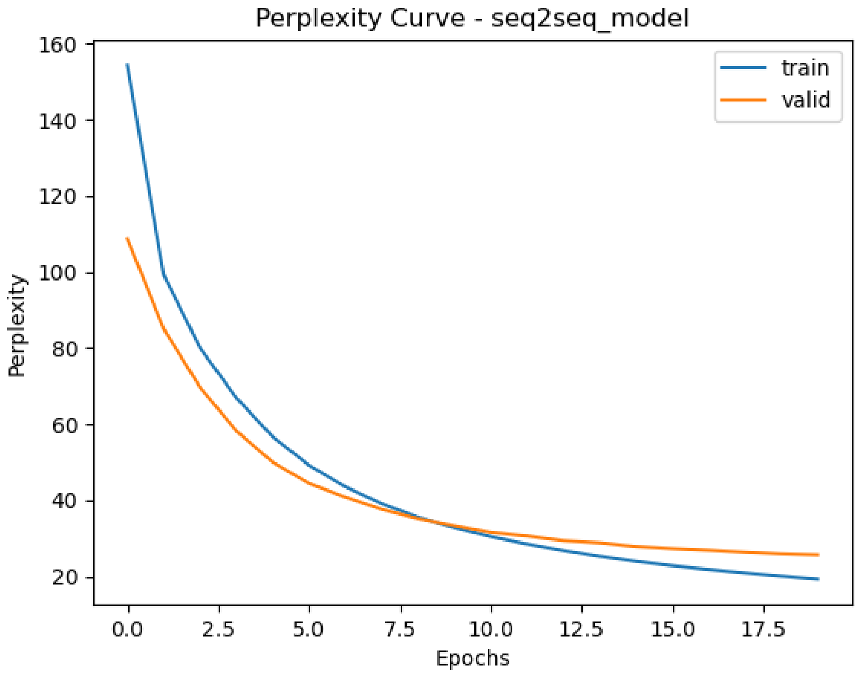
List your best model hyper-parameter values including model type: ?

encoder_emb_size = 128, encoder_hidden_size = 128, encoder_dropout = 0.3
decoder_emb_size = 128, decoder_hidden_size = 128, decoder_dropout = 0.3

learning_rate = 2*1e-3, model_type = "LSTM"

EPOCHS = 20

Seq2Seq Best model Learning Curves (Perplexity)



Seq2Seq Explanation – Best model

Explain the details of your best model. Explain what you did to improve your model’s performance and why

From previous experiment, we know that LSTM models outperform vanily RNNs significantly. In additiona, attention mechanism really help the models focus on important inputs, help reduce overfitting and better performance.
It is obvious that we start out with LSTM models with attention.
There is still small gap between training and validation so there is small overfitting and it is hard to get rid of overfitting completely so we start by increasing dropout, hopefully help decrease overfitting. We stop at dropout = 0.3
Second hyperparameter have a lot of impact is learning rate. We change this together with epoch to find optimal one. Lower learning rate can help improve performance but might require more iteration (increase epoch). We stops when learning rate is 0.002 and epoch is 20. This is not too fast but not too slow for good result.
Lastly, we did tune the embedding and hidden size but the improvement is not significant, sometime we even have worse results, due to this hidden dimension is remained at 128. Increasing hidden dimension does require higher computation power (higher training time)

Transformer Results
Values are for last epoch

Default configuration (Encoder Only)
Training Loss: 2.1527
Training Perplexity: 8.6082
Validation Loss: 3.1140
Validation Perplexity: 22.5410

Default configuration (full transformer)
Training Loss: 0.9980
Training Perplexity: 2.7128
Validation Loss: 1.8274
Validation Perplexity: 6.2177

Best model (full transformer)
Training Loss: 1.2572
Training Perplexity: 3.5156
Validation Loss: 1.7531
Validation Perplexity: 5.7725

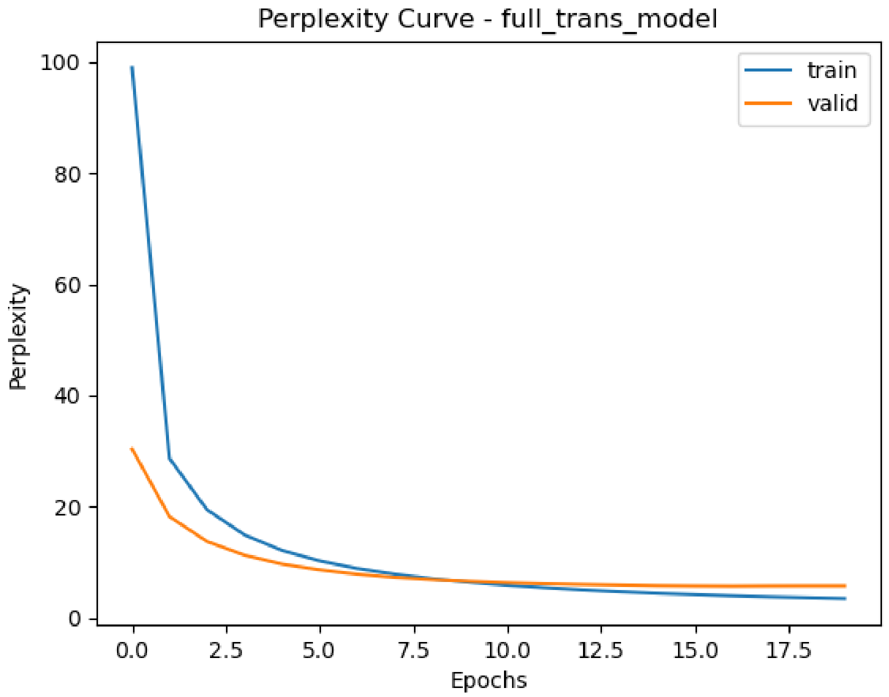
List your best model hyper-parameter values (full transformer):
Output_size = 5, hidden dim = 128

Number_heads = 4, dim_feedforward = 2048

Learning rate = 5*1e-4

Epochs = 20

Full Transformer Best model Learning Curves (Perplexity)



Full Transformer Explanation – Best model

Explain what you did here and why you did it to improve your model performance.

First of all, the model result is much better than RNN or LSTM. From our experiment, the full transformer model outperform transformer with encoder only. Therefore we started out with full transformer model to tune.
We did tune hidden dimension but it seem performance change is insignificant and sometime worse so we keep it at 128. Training time is also higher with higher hidden dimension as well so we keep it low if it is not necessarily improving.

We tune the number of heads or multi-head attention. By increasing num_head to 4 from 2, we does see improvement in perplexity score. Because the head learns different perspective in the sequence, increase it result in better feature extraction. It also capture more dependencies in data.
Again, we tune learning rate because it is very sensitive. Lower learning rate does result in better perplexity score but require more training time, higher imitation so increase epoch. At some points, increasing epoch lead to overfitting, when training curve decrease and validation curve no longer better.
After some tuning, we decrease the learning rate by half while keeping epoch at 20 and have a better results.
We also tested it in translation and the results are somewhat ok, not perfect but most sentences are very close.

Transformer (Encoder Only) Translation Results (default settings)

Put translation results for your model (1st 9 sentences) here. You may remove duplicate <pad> and <eos> tokens for each sentence.

True Translation	Predicted Translation
'<sos>', 'a', 'man', 'in', 'an', 'orange', 'hat', 'starring', 'at', 'something', '<eos>'	'<sos>', 'a', 'man', 'in', 'an', 'orange', 'hat', 'hat', 'something', 'something', 'something', '<eos>’
'<sos>', 'a', 'boston', 'terrier', 'is', 'running', 'on', 'lush', 'green', 'grass', 'in', 'front', 'of', 'a', 'white', 'fence', '<eos>'	'<sos>', 'a', 'boston', 'vendor', 'runs', 'walking', 'the', 'green', 'grass', 'grass', 'grass', 'of', 'of', 'fence', 'fence', 'fence', '<eos>'
'<sos>', 'a', 'girl', 'in', 'karate', 'uniform', 'breaking', 'a', 'stick', 'with', 'a', 'front', 'kick', '<eos>'	'<sos>', 'a', 'girl', 'in', 'a', 'a', 'a', 'with', 'a', 'a', 'a', 'a', '<eos>
'<sos>', 'five', 'people', 'wearing', 'winter', 'jackets', 'and', 'helmets', 'stand', 'in', 'the', 'snow', 'with', '<unk>', 'in', 'the', 'background', '<eos>'	'<sos>', 'five', 'people', 'in', 'in', 'and', 'helmets', 'helmets', 'helmets', 'in', 'in', 'snow', 'in', 'the', 'in', 'in', '<eos>', 'background', 'background', '<eos>'
'<sos>', 'people', 'are', 'fixing', 'the', 'roof', 'of', 'a', 'house', '<eos>'	'<sos>', 'people', 'are', 'the', 'the', 'a', 'a', '<eos>'
'<sos>', 'a', 'group', 'of', 'people', 'standing', 'in', 'front', 'of', 'an', 'igloo', '<eos>'	'<sos>', 'a', 'group', 'of', 'people', 'standing', 'standing', 'in', 'front', '<eos>’
'<sos>', 'a', 'guy', 'works', 'on', 'a', 'building', '<eos>'	'<sos>', 'a', 'guy', 'is', 'on', 'a', 'building', 'building', '<eos>'
'<sos>', 'a', 'man', 'in', 'a', 'vest', 'is', 'sitting', 'in', 'a', 'chair', 'and', 'holding', 'magazines', '<eos>'	'<sos>', 'a', 'man', 'in', 'a', 'vest', 'vest', 'on', 'on', 'a', 'chair', 'a', 'and', '<eos>'
'<sos>', 'a', 'mother', 'and', 'her', 'young', 'song', 'enjoying', 'a', 'beautiful', 'day', 'outside', '<eos>'	'<sos>', 'a', 'mother', 'and', 'her', 'small', 'son', 'enjoying', 'a', '<eos>'

Table 1

Full Transformer Translation Results (best model)

Put translation results for your best model (1st 9 sentences) here. You may remove duplicate <pad> and <eos> tokens for each sentence.

True Translation	Predicted Translation
'<sos>', 'a', 'man', 'in', 'an', 'orange', 'hat', 'starring', 'at', 'something', '<eos>'	'<sos>', 'a', 'man', 'in', 'an', 'orange', 'hat', 'is', 'carving', 'something', '<eos>'
'<sos>', 'a', 'boston', 'terrier', 'is', 'running', 'on', 'lush', 'green', 'grass', 'in', 'front', 'of', 'a', 'white', 'fence', '<eos>'	'<sos>', 'a', 'boston', 'member', 'runs', 'across', 'the', 'green', 'grass', 'in', 'front', 'of', 'a', 'white', 'fence', '<eos>'
'<sos>', 'a', 'girl', 'in', 'karate', 'uniform', 'breaking', 'a', 'stick', 'with', 'a', 'front', 'kick', '<eos>'	'<sos>', 'a', 'girl', 'in', 'a', 'karate', 'uniform', 'is', 'teaching', 'a', 'board', 'with', 'a', 'kick', '<eos>'
'<sos>', 'five', 'people', 'wearing', 'winter', 'jackets', 'and', 'helmets', 'stand', 'in', 'the', 'snow', 'with', '<unk>', 'in', 'the', 'background', '<eos>'	'<sos>', 'five', 'people', 'in', 'traditional', 'outfits', 'and', 'helmets', 'are', 'standing', 'in', 'the', 'snow', 'with', 'snow', 'in', 'the', 'background', '<eos>'
'<sos>', 'people', 'are', 'fixing', 'the', 'roof', 'of', 'a', 'house', '<eos>'	'<sos>', 'people', 'fix', 'the', 'roof', 'of', 'a', 'house', '<eos>'
'<sos>', 'a', 'group', 'of', 'people', 'standing', 'in', 'front', 'of', 'an', 'igloo', '<eos>'	'<sos>', 'a', 'group', 'of', 'people', 'standing', 'in', 'front', 'of', 'a', 'outdoor', 'venue', '<eos>'
'<sos>', 'a', 'guy', 'works', 'on', 'a', 'building', '<eos>'	'<sos>', 'a', 'guy', 'working', 'on', 'a', 'building', '<eos>'
'<sos>', 'a', 'man', 'in', 'a', 'vest', 'is', 'sitting', 'in', 'a', 'chair', 'and', 'holding', 'magazines', '<eos>'	'<sos>', 'a', 'man', 'in', 'a', 'vest', 'is', 'sitting', 'on', 'a', 'chair', 'holding', 'books', '<eos>'
'<sos>', 'a', 'mother', 'and', 'her', 'young', 'song', 'enjoying', 'a', 'beautiful', 'day', 'outside', '<eos>'	'<sos>', 'a', 'mother', 'and', 'her', 'little', 'girl', 'enjoy', 'a', 'nice', 'day', 'outdoors', '<eos>'

Table 2

Seq2Seq (Best model) Translation Results

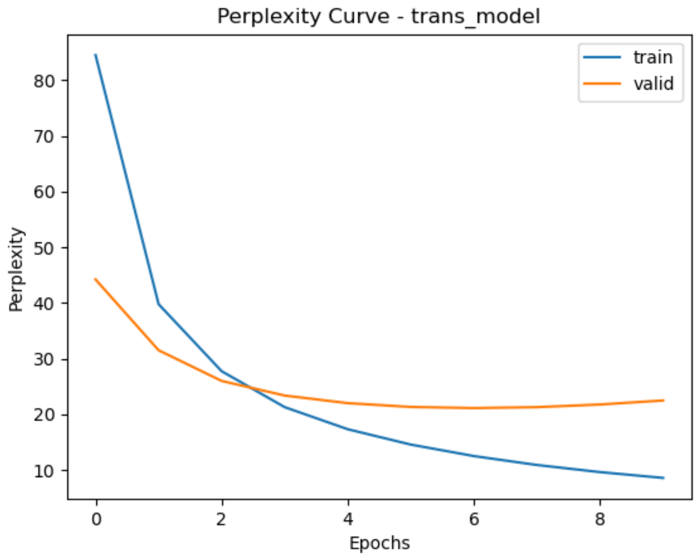
Put translation results for your best model (1st 9 sentences) here. You may remove duplicate <pad> and <eos> tokens for each sentence.

True Translation	Predicted Translation
'<sos>', 'a', 'man', 'in', 'an', 'orange', 'hat', 'starring', 'at', 'something', '<eos>'	'<sos>', 'a', 'man', 'with', 'a', 'hat', 'hat', 'is', 'his', '<eos>'
'<sos>', 'a', 'boston', 'terrier', 'is', 'running', 'on', 'lush', 'green', 'grass', 'in', 'front', 'of', 'a', 'white', 'fence', '<eos>'	'<sos>', 'a', 'large', 'horse', 'is', 'is', 'a', 'a', 'a', 'a', 'a', 'white', 'a', '<eos>'
'<sos>', 'a', 'girl', 'in', 'karate', 'uniform', 'breaking', 'a', 'stick', 'with', 'a', 'front', 'kick', '<eos>'	'<sos>', 'a', 'girl', 'in', 'a', 'a', 'outfit', 'a', 'a', 'a', 'a', 'a', '<eos>'
'<sos>', 'five', 'people', 'wearing', 'winter', 'jackets', 'and', 'helmets', 'stand', 'in', 'the', 'snow', 'with', '<unk>', 'in', 'the', 'background', '<eos>'	'<sos>', 'five', 'men', 'in', 'helmets', 'and', 'and', 'in', 'in', 'in', 'in', 'in', 'in', 'in', 'in', 'background', '<eos>'
'<sos>', 'people', 'are', 'fixing', 'the', 'roof', 'of', 'a', 'house', '<eos>'	'<sos>', 'people', 'are', 'the', 'the', 'of', 'a', 'of', '<eos>'
'<sos>', 'a', 'group', 'of', 'people', 'standing', 'in', 'front', 'of', 'an', 'igloo', '<eos>'	'<sos>', 'a', 'group', 'of', 'people', 'are', 'standing', 'a', 'a', 'of', '<eos>'
'<sos>', 'a', 'guy', 'works', 'on', 'a', 'building', '<eos>'	'<sos>', 'a', 'guy', 'is', 'working', 'on', 'a', 'a', '<eos>'
'<sos>', 'a', 'man', 'in', 'a', 'vest', 'is', 'sitting', 'in', 'a', 'chair', 'and', 'holding', 'magazines', '<eos>'	'<sos>', 'a', 'man', 'in', 'a', 'suit', 'suit', 'is', 'sitting', 'a', 'chair', 'chair', '<eos>'
'<sos>', 'a', 'mother', 'and', 'her', 'young', 'song', 'enjoying', 'a', 'beautiful', 'day', 'outside', '<eos>'	'<sos>', 'a', 'mother', 'and', 'a', 'daughter', 'enjoying', 'at', 'at', 'at', 'at', '<eos>'

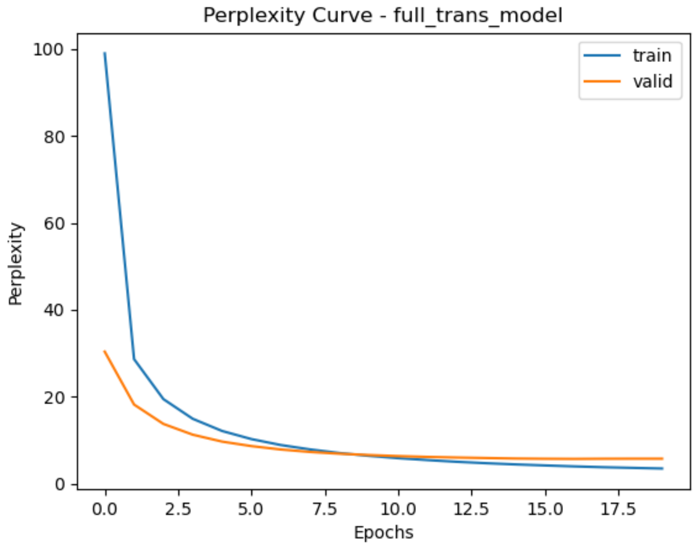
Table 3

Compare Transformer (Encoder Only) to Transformer (Full transformer)

Compare your results for default settings for Encoder Only Transformer vs best model for Full transformer both quantitatively and qualitatively. Explain why you see differences.



Encoder only transformer



Full transformer

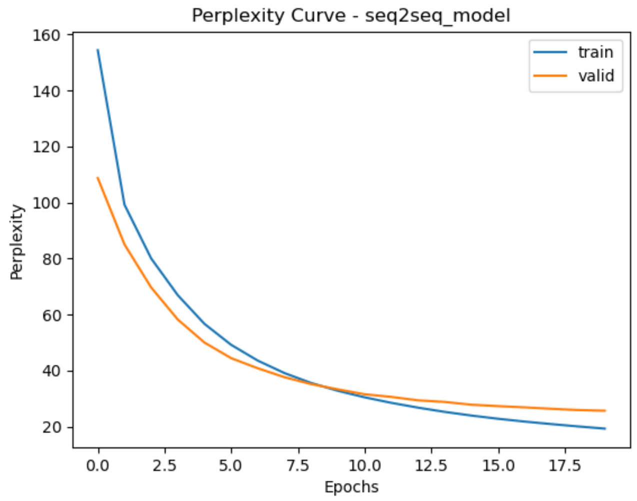
Compare Transformer (Encoder Only) to Transformer (Full transformer)

Compare your results for default settings for Encoder Only Transformer vs best model for Full transformer both quantitatively and qualitatively. Explain why you see differences.

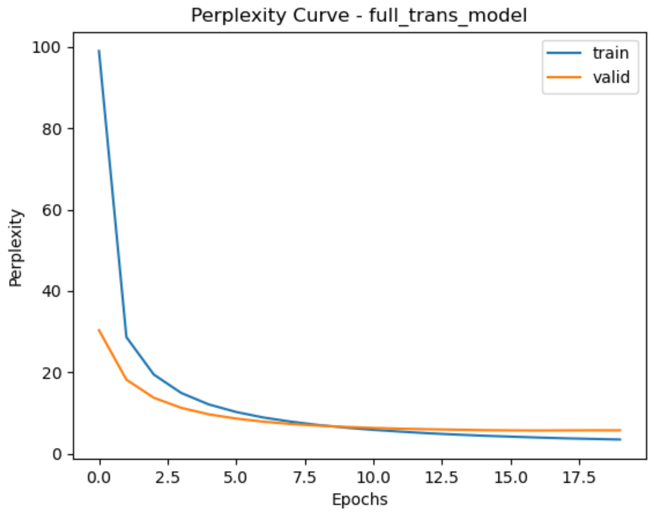
Encoder only transformer validation perplexity is 22 while full transformer is 5, indicating much better performance in full transformer. The encoder only version has validation curve no longer improve while training curve still improving, indicating overfitting.
Both model performance improve after only a few epochs (easy convergence)
The reason for worse performance in encoder only is because it lack decoder. So encoding task help with classification or encoding task, but struggle with generative tasks like in this case translation so it need decoder.
In encoder, model rely only on attention within input tokens. While in full transformer, it has cross attention between encoder and decoder, so better for translation task.
Both models are good but full transformer outperform, generalize better and less overfitting.

Compare Seq2Seq to Transformer (Best models)

Compare your Seq2Seq best model results to your Transformer best model results both quantitatively and qualitatively and explain the differences.



Seq2seq LSTM w/ attention



Full transformer

Compare Seq2Seq to Transformer (Best models)

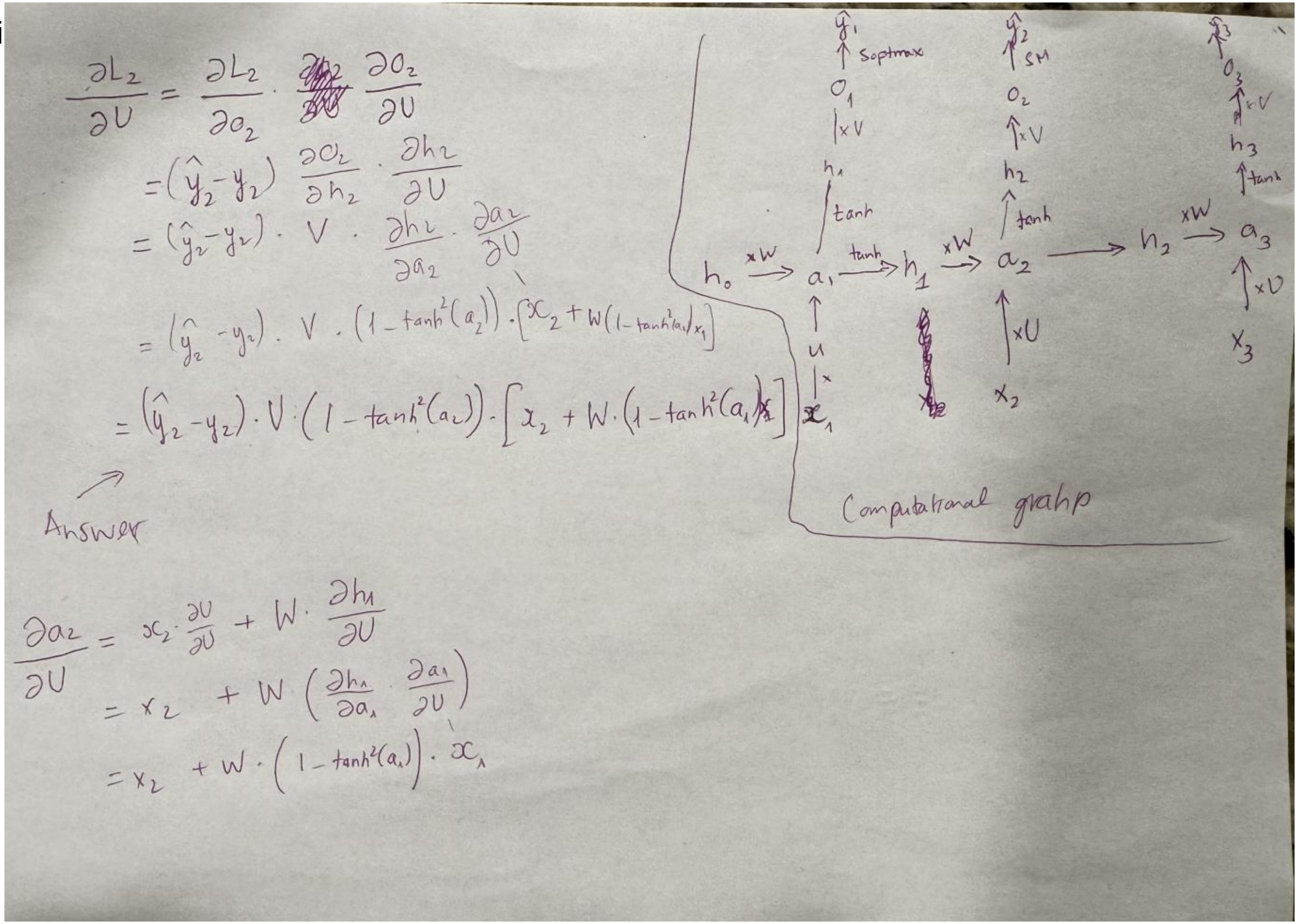
Compare your Seq2Seq best model results to your Transformer best model results both quantitatively and qualitatively and explain the differences.

Both models perform well but full transformer achieve much lower validation perplexity (good).
The full transformer converge faster after a few epoch (after 2 poch vs 5)
The gap between training and validation curve is wider in seq2seq model (LSTM with attention) so seq2seq is more overfitting.
Both models seem to generalize very well.
In term of time, even though the full transformer have more parameters but it train faster compared to seq2seq. This is due to self-attention can be parallel training in full transformer. training, full transformer is more efficient.
LSTM can still subject to vanishing gradient, so long range dependencies is worse than full transformer.

Overall, full transformer outperform seq2seq in all aspects. It might require more training time but if computation allow parallel training then it can be better.

Theory question

Add additi



$$\frac{\partial L_2}{\partial U} = \frac{\partial L_2}{\partial o_2} \cdot \cancel{\frac{\partial o_2}{\partial U}} \; \frac{\partial o_2}{\partial U}$$

$$= (\hat{y}_2 - y_2) \; \frac{\partial o_2}{\partial h_2} \cdot \frac{\partial h_2}{\partial U}$$

$$= (\hat{y}_2 - y_2) \cdot V \cdot \frac{\partial h_2}{\partial a_2} \cdot \frac{\partial a_2}{\partial U}$$

$$= (\hat{y}_2 - y_2) \cdot V \cdot (1 - \tanh^2(a_2)) \cdot [x_2 + W(1 - \tanh^2(a_1)) x_1]$$

$$= (\hat{y}_2 - y_2) \cdot V \cdot (1 - \tanh^2(a_2)) \cdot [x_2 + W \cdot (1 - \tanh^2(a_1)\cancel{x})]$$

↗ Answer

$$\frac{\partial a_2}{\partial U} = x_2 \cdot \frac{\partial U}{\partial U} + W \cdot \frac{\partial h_1}{\partial U}$$

$$= x_2 + W \left(\frac{\partial h_1}{\partial a_1} \cdot \frac{\partial a_1}{\partial U} \right)$$

$$= x_2 + W \cdot (1 - \tanh^2(a_1)) \cdot x_1$$

$$\begin{tikzcd}
 & \hat{y}_1 & & \hat{y}_2 & & \hat{y}_3 \\
 & O_1 \arrow[u, "\text{Soptmax}"'] & & O_2 \arrow[u, "\text{SM}"'] & & O_3 \arrow[u] \\
 & h_1 \arrow[u, "\times V"'] & & h_2 \arrow[u, "\times V"'] & & h_3 \arrow[u, "\times V"'] \\
h_0 \arrow[r, "\times W"] & a_1 \arrow[u, "\tanh"'] \arrow[r, "\tanh"] & h_1 \arrow[r, "\times W"] & a_2 \arrow[u, "\tanh"'] \arrow[r] & h_2 \arrow[r, "\times W"] & a_3 \arrow[u, "\tanh"'] \\
 & u \arrow[u] & & & & \\
 & x_1 \arrow[u, "\times"'] & & x_2 \arrow[uu, "\times U"'] & & x_3 \arrow[uu, "\times U"']
\end{tikzcd}$$

Computational grahp

Paper discussion – Short review

Language Models are Few-Shot Learners:
Language Models usually require thousands of training examples or fine-tuning before achieve good results. Instead, the paper introduce a GPT-3 that are trained with significantly more parameters than other non-sparse language model and achieve good result in few-shot settings. A scale up language model like GPT3 can solve task like translation, reading comprehension, arithmetic maths, etc. without fine-tuning or in few-shot settings. I can also generate human-like text that are close to what human can do. This is great because fine-tuning require label data and it is costly. Near the end, the papers raised ethical concerns of misage, as well as biases because the model is trained using internet data.

Some strength the paper pointed out is that, NLP model can improve performance in few-shot settings by increasing model size/upscale. It can also performance well with minimal training example and efficient because require less label data. Performance in variety of area equal if not better than human.
The weakness is that the upscale model require pre-training that are costly like GPT3 with 175 billion parameters is huge. The model is subject to bias due to the data came from internet and not all tasks can perform well.

My personal take away is that a single common NLP model can be used to solve different tasks without fine-tuning that is costly. Instead, the model can be trained and generalize across different task. This can be time saving and efficient in some ways. Although, the models does need more example compared to human to learn new task. Lastly, it opens question about ethical concern and biases and if there is a way to train to improve data used to train.

Answer specific question

In section 5, limitation, couple of weaknesses still appear in GPT3. They are text synthesis task, answer in discrete language task, in-context learning performance has gaps like evaluating different words with same meaning or implications. It is point out that this is structural and algorithmic limitation. Result of the experiements do not include bidirectional architectures or training objective like denoising. So we can try to include or make bidirectional model that works in few-shot settings.
Another limit is lack pretraining objective. This can be improved by include learning objective function from human, fine-tune with reinforcement learning or adding additional modalities.
The poor sample efficiency during pre-training can be solved by using improve pre-training sample, providing additional information.
Another limitation is GPT3 is both expensive and inconvenient to performance inference on. Possible direction is using distillation of large models down to a manageable size for specific task
There are couple of social implications of these models. Many tasks like chatbots, writing assistance can help improve productivity. Generating image caption can help users with visual impair. Other implication worth noting is learning from AI, summarization tools, or translation.
Other implication might not be good including biases and harmful stereotypes results due to biased data used to train. Ethical concerns are raised about scams and fraud came from deceptive chatbots. This could also open more discussion about building an ethical guidelines, etc.