Assignment 3 (NLP) Writeup

DO NOT TAG

Name: Nicolas Gonzalez
GT Email: ngonzalez62@gatech.edu

Seq2Seq Results – Default configuration

Values are for last epoch

RNN

Training Loss: 4.6135
Training Perplexity: 100.8337
Validation Loss: 4.7032
Validation Perplexity: 110.2951

LSTM

Training Loss: 3.2009
Training Perplexity: 24.5554
Validation Loss: 3.3634
Validation Perplexity: 28.8879

RNN-with-Attention

Training Loss: 3.2975
Training Perplexity: 27.0454
Validation Loss: 3.4313
Validation Perplexity: 30.9181

LSTM-with-Attention

Training Loss: 3.1733
Training Perplexity: 23.8861
Validation Loss: 3.3617
Validation Perplexity: 28.8391

Seq2Seq Explanation (RNN vs LSTM)

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

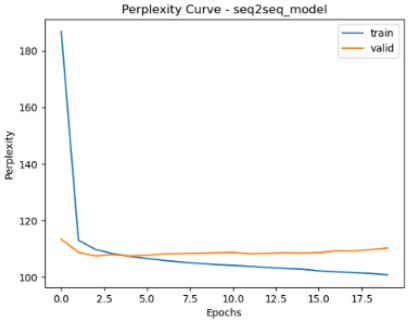
LSTMs perform better than RNNs on all collected metrics: loss, perplexity, training, validation, with attention, without attention.

This makes sense, as RNNs were developed precisely to address some of the well-known shortcomings of RNNs.

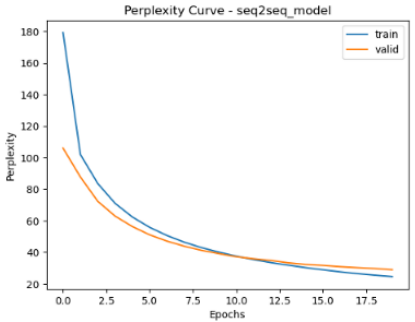
One well known issue of RNNs is that they tend to suffer from the exploding or vanishing gradient problem. As we saw in lecture, this happens because the RNN architecture causes gradients to explode or vanish depending on whether the norm of the weight matrix is greater or less than 1. We can see in the RNN perplexity curves that the RNN learns quickly in the first few epochs, and then the learning flattens out. That is when the gradients start to vanish. With the LSTM architecture, we see much smoother perplexity curves. LSTMs don't suffer as much from vanishing/exploding gradients because they were architected to have additive updates to the cell state instead of relying purely on multiplicative updates (as vanilla RNNs do).

We can also see that after 2 iterations, the validation perplexity curve of the RNN starts to increase. This is a sign of overfitting. This likely happens because RNNs collapse all read information into a single vector, without intelligently highlighting the parts of the sentence that are most relevant to predicting the next token. This likely makes the RNN rely on short sequences of words that appear often in the training data, but do not generalize well. LSTMs mitigate this issue by designing a system that allows the prediction of the next word to look at its short AND long term memory, to see which parts of the previous text are most helpful for predicting the next word.

RNN, no attention:



LSTM, no attention:



Seq2Seq Explanation (RNN vs RNN-with-Attention)

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

Out of the four main combinations (RNN, LSTM, RNN-with-Attention, LSTM-with-Attention), the last 3 all have similar performance. Looking at validation perplexity, we see values of 28.89, 30.92, 28.84. But RNN without attention has a validation perplexity of 110. That is **much** worse.

In last slide we already discussed how replacing an RNN with an LSTM helps. However, what is interesting to note here is that augmenting the RNN with attention helps about as much.

The reason that attention helps so much is that without attention, we run through the encoder, generate the output and the hidden state, but then *never look at the encoder output again*. We rely entirely on the encoder hidden state for decoding.

But attention's improved performance demonstrates that the encoder output offers a lot of valuable information. Attention looks at the entire encoder output, weights it by similarity to the next token to be predicted, and that allows the decoder to look at precisely the parts of the german text that are most relevant to predicting the next English word. This gets rid of RNN's biggest limitation of having to encode the entire meaning of a full sentence into a single vector.

Seq2Seq Results – Best model

Values are for last epoch

Your best model after hyper-parameter tuning

Best model

Training Loss: 2.7442

Training Perplexity: 15.5526

Validation Loss: 3.1981

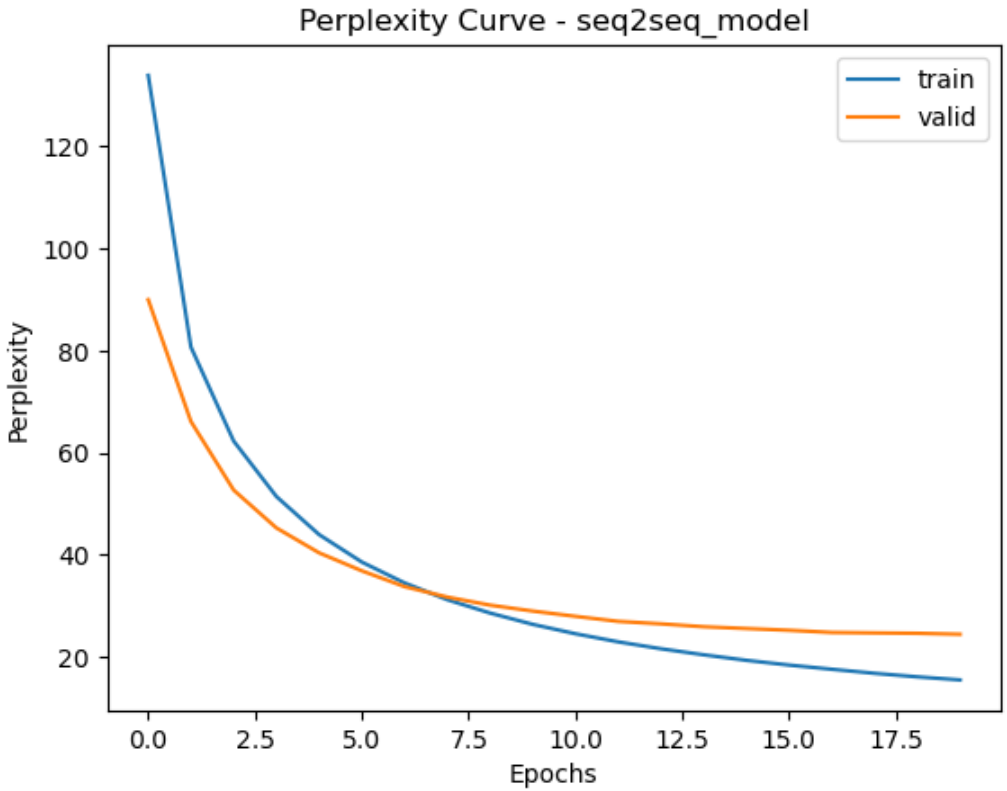
Validation Perplexity: 24.4853

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

BATCH_SIZE = 64

encoder_emb_size = 128
encoder_hidden_size = 150
encoder_dropout = 0.2
decoder_emb_size = 128
decoder_hidden_size = 150
decoder_dropout = 0.2
learning_rate = 0.001
model_type = "LSTM"
EPOCHS = 20
attention = True

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.

Since this model is slow to train, and since there are over 10 possible hyperparameters to tune, I decided not to do a grid-search, as that would yield too many possible hyperparameter combinations.

Instead, I decided to tune hyperparameters based on experience and intuition, starting with the hyperparameters that, in my experience, have the biggest impact, and moving on to hyperparameters that tend to be less relevant.

In my experience, the number of epochs has a big impact on model performance. However, the assignment explicitly says that simply tuning epochs is not interesting. Because of this (and because of the impact on training time), I decided to leave epochs at 20, and try to squeeze out the best possible performance by tuning other hyperparameters.

With limited epochs, learning rate is an important hyper parameter because it dictates how big the "steps" down the loss landscape are. The default was 0.001. I tried a bit higher (0.005) and a bit lower (0.0005), and both resulted in worsened validation loss. The higher learning rate was too big, likely "overstepping" descents in the loss landscape. The lower learning rate took steps that were too small, and in just 20 epochs, could not reach as low of a point in the loss landscape.

I then tuned batch size, which in my experience, can really help when the number of epochs is limited. This is because a smaller batch size means more iterations per epoch, and theremore more steps down the loss landscape. The default batch size was 128. I tried 64, which improved validation loss. Going as low as 32 worsened the validation loss. At that low of a batch size, the batch is no longer a good enough proxy for the full training set.

I then tuned encoder and decoder dropout in unison. I found that decreasing and increasing dropout (relative to the default 0.2) worsened validation loss. Decreasing it led to a bit of overfitting (evident in the perplexity curves), and increasing it led to underfitting.

I then tried to tune model capacity (by tuning the hidden size of the encoder and decoder in unison). I found that a mild increase from 128 to 150 improved validation loss. A more significant increase in model capacity worsened validation loss, likely because there amount of epochs and data were insufficient to tune a larger capacity model.

Lastly, I tried to tune the encoder and decoder embedding size. I found that increasing and decreasing both worsened validation loss. This means that the default embedding size of 128 was appropriate for the vocabulary size.

Transformer Results

Values are for last epoch

Default configuration (Encoder Only)

Training Loss: 2.1523
Training Perplexity: 8.6050
Validation Loss: 3.1136
Validation Perplexity: 22.5030

Default configuration (full transformer)

Training Loss: 1.4964
Training Perplexity: 4.4654
Validation Loss: 1.8474
Validation Perplexity: 6.3435

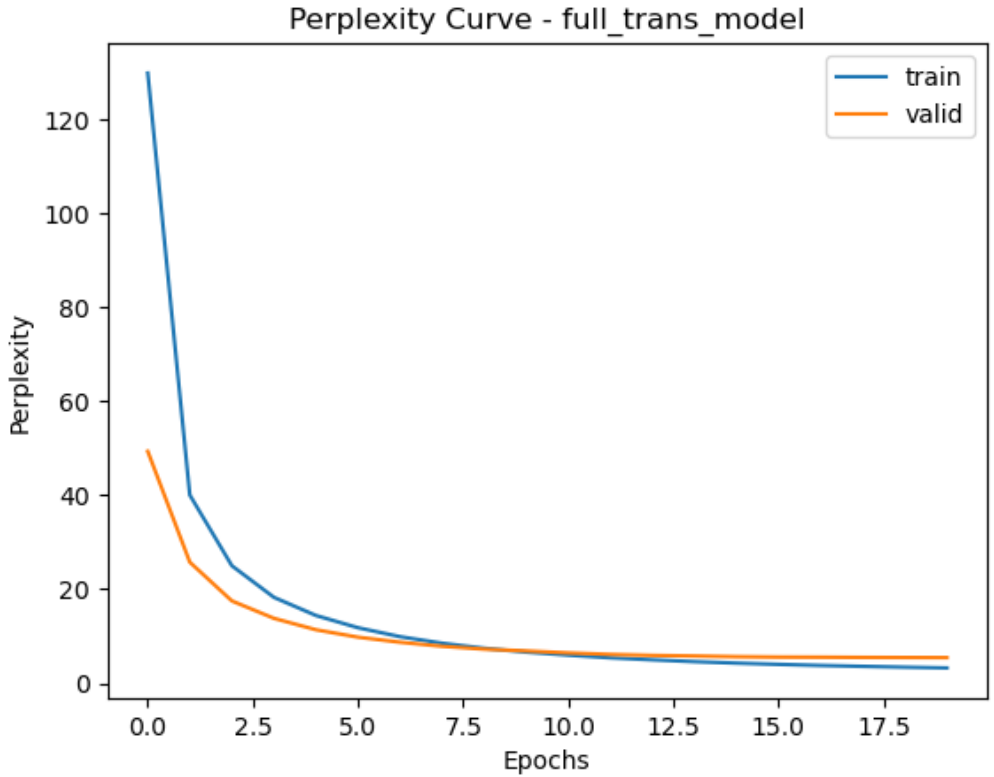
Best model (full transformer)

Training Loss: 1.1759
Training Perplexity: 3.2411
Validation Loss: 1.6925
Validation Perplexity: 5.4331

List your best model hyper-parameter values (full transformer):

Batch_size = 256
learning_rate = 0.00075
EPOCHS = 20
hidden_dim=200
num_heads=1
dim_feed_forward=512
num_layers_enc=4
num_layers_dec=4
dropout=0.2

Full Transformer Best model Learning Curves (Perplexity)



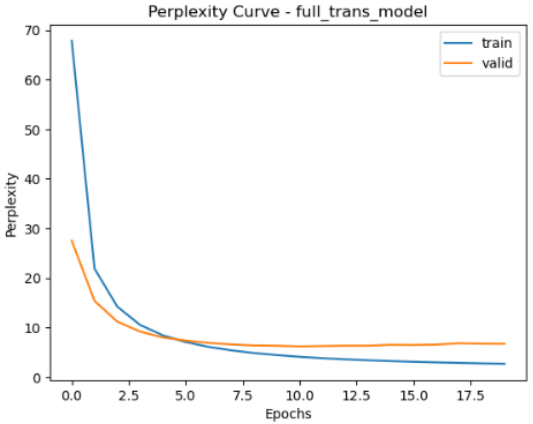
Full Transformer Explanation – Best model

Explain what you did here and why you did it to improve your model performance.
When I tuned the Seq2Seq model, I held the number of epochs constant at 20. So, I decided to also use 20 epochs for the Full Transformer. The default was set to 10 epochs, and increasing that to 20 was enough to demonstrate overfitting (notice on the figure to the right, that perplexity of the validation decreases and then starts increasing).

To address the overfitting, I tried increasing dropout since it is specifically meant to mitigate overfitting. What I found is that higher dropout got rid of the of the increasing pattern in the validation curve, but worsened the achieved perplexity. It is likely that with more data and more epochs, the model would be able to tolerate higher dropout.

The next thing I tried was to gradually increase the batch size. My hope was that with more examples in each batch, the model would be less likely to overfit to any specific example. I found that this helped a little bit, and so I settled on a batch size of 256. Higher batch sizes than that resulted in too few iterations per epoch, and therefore insufficient epochs.

Full transformer, default hyperparameters except with 20 epochs instead of 10:



At this point, I recalled from lecture that it often makes sense to change batch size and learning rate in tandem. The idea here is that if the batch size is larger, the batch's approximation of the loss landscape will be more accurate, and therefore larger "steps" are tolerable. What I found surprised me: my model improved with smaller, not larger learning rate. My interpretation of this is that the default learning rate was a bit big for the complexity of the loss landscape. A learning rate of 0.00075 improved my achieved validation loss and perplexity.

My next step was to try increasing and decreasing each of hidden_dims, num_heads, dim_feed_forward, num_layers_enc, and num_layers_dec. What I found is that I was able to achieve the best performance by increasing hidden_dims from 128 to 200, decreasing the num_heads from 2 to 1, decreasing the dim_feed_forward from 2048 to 512, and increasing the number of layers of the encoders from 2 to 4. This is somewhat surprising to me, because I imagined all of these hyperparameters as representing "capacity", and imagined that an optimal configuration would bring all of them up, or all of them down. What this result suggests is that model capacity is not a single-dimensional concept, and different components of the Transformer contribute to expressiveness in different ways. Increasing depth (more layers) and widening the hidden dimension likely helped the model capture richer sequential dependencies, while reducing the number of heads and the feed-forward size may have prevented it from spreading its attention too thin or overfitting small patterns. In other words, the final configuration seems to provide just enough representational capacity in the most impactful parts of the network, while avoiding excessive complexity in others.

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>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>'	'<sos>', 'a', 'man', 'in', 'an', 'hat', 'hat', 'hat', 'something', 'something', 'something', 'something', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>'
'<sos>', 'a', 'boston', 'terrier', 'is', 'running', 'on', 'lush', 'green', 'grass', 'in', 'front', 'of', 'a', 'white', 'fence', '<eos>', '<pad>', '<pad>', '<pad>'	'<sos>', 'a', 'boston', 'vendor', 'runs', 'through', 'grass', 'grass', 'grass', 'grass', 'of', 'of', 'of', 'fence', 'fence', 'fence', '<eos>', '<eos>', '<eos>', '<eos>'
'<sos>', 'a', 'girl', 'in', 'karate', 'uniform', 'breaking', 'a', 'stick', 'with', 'a', 'front', 'kick', '<eos>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>'	'<sos>', 'a', 'girl', 'in', 'a', 'a', 'a', 'with', 'a', 'a', 'a', 'a', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>'
'<sos>', 'five', 'people', 'wearing', 'winter', 'jackets', 'and', 'helmets', 'stand', 'in', 'the', 'snow', 'with', '<unk>', 'in', 'the', 'background', '<eos>', '<pad>', '<pad>'	'<sos>', 'five', 'people', 'in', 'in', 'and', 'with', 'helmets', 'stand', 'in', 'the', 'snow', 'in', 'the', 'in', 'in', '<eos>', 'background', '<eos>', '<eos>'
'<sos>', 'people', 'are', 'fixing', 'the', 'roof', 'of', 'a', 'house', '<eos>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>'	'<sos>', 'people', 'are', 'the', 'the', 'of', 'a', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>'
'<sos>', 'a', 'group', 'of', 'people', 'standing', 'in', 'front', 'of', 'an', 'igloo', '<eos>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>'	'<sos>', 'a', 'group', 'of', 'people', 'standing', 'standing', 'in', 'a', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>'
'<sos>', 'a', 'guy', 'works', 'on', 'a', 'building', '<eos>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>'	'<sos>', 'a', 'guy', 'is', 'on', 'a', 'building', 'building', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>'
'<sos>', 'a', 'man', 'in', 'a', 'vest', 'is', 'sitting', 'in', 'a', 'chair', 'and', 'holding', 'magazines', '<eos>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>'	'<sos>', 'a', 'man', 'in', 'a', 'vest', 'vest', 'on', 'on', 'a', 'chair', 'a', 'holding', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>'
'<sos>', 'a', 'mother', 'and', 'her', 'young', 'song', 'enjoying', 'a', 'beautiful', 'day', 'outside', '<eos>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>'	'<sos>', 'a', 'mother', 'and', 'her', 'small', 'son', 'enjoying', 'a', '<eos>', 'day', 'day', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<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>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>'	'<sos>', 'a', 'man', 'in', 'an', 'orange', 'hat', 'carving', 'something', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>'
'<sos>', 'a', 'boston', 'terrier', 'is', 'running', 'on', 'lush', 'green', 'grass', 'in', 'front', 'of', 'a', 'white', 'fence', '<eos>', '<pad>', '<pad>', '<pad>'	'<sos>', 'a', 'boston', '<unk>', 'is', 'running', 'across', 'a', 'green', 'grass', 'in', 'front', 'of', 'a', 'white', 'fence', '<eos>', '<eos>', '<eos>', '<eos>'
'<sos>', 'a', 'girl', 'in', 'karate', 'uniform', 'breaking', 'a', 'stick', 'with', 'a', 'front', 'kick', '<eos>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>'	'<sos>', 'a', 'girl', 'in', 'a', 'karate', 'suit', 'is', 'using', 'a', 'board', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>'
'<sos>', 'five', 'people', 'wearing', 'winter', 'jackets', 'and', 'helmets', 'stand', 'in', 'the', 'snow', 'with', '<unk>', 'in', 'the', 'background', '<eos>', '<pad>', '<pad>'	'<sos>', 'five', 'people', 'in', 'snow', 'with', 'helmets', 'and', 'helmets', 'are', 'standing', 'in', 'the', 'background', '<eos>', 'in', 'the', 'background', '<eos>', '<eos>'
'<sos>', 'people', 'are', 'fixing', 'the', 'roof', 'of', 'a', 'house', '<eos>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>'	'<sos>', 'people', 'repair', 'the', 'roof', 'of', 'a', 'house', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>'
'<sos>', 'a', 'group', 'of', 'people', 'standing', 'in', 'front', 'of', 'an', 'igloo', '<eos>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>'	'<sos>', 'a', 'group', 'of', 'people', 'stand', 'in', 'front', 'of', 'a', 'regal', 'shop', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>'
'<sos>', 'a', 'guy', 'works', 'on', 'a', 'building', '<eos>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>'	'<sos>', 'a', 'guy', 'working', 'on', 'a', 'building', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>'
'<sos>', 'a', 'man', 'in', 'a', 'vest', 'is', 'sitting', 'in', 'a', 'chair', 'and', 'holding', 'magazines', '<eos>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>'	'<sos>', 'a', 'man', 'in', 'a', 'vest', 'is', 'sitting', 'on', 'a', 'chair', 'holding', 'basketballs', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>'
'<sos>', 'a', 'mother', 'and', 'her', 'young', 'song', 'enjoying', 'a', 'beautiful', 'day', 'outside', '<eos>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>'	'<sos>', 'a', 'mother', 'and', 'her', 'young', 'son', 'enjoying', 'a', 'beautiful', 'day', 'outside', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>'

Table 2

Seq2Seq (Best model) Translation Results

Put translation results for your best model (1 st 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>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>'	'<sos>', 'a', 'man', 'with', 'a', 'hat', 'hat', 'is', 'his', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>'
'<sos>', 'a', 'boston', 'terrier', 'is', 'running', 'on', 'lush', 'green', 'grass', 'in', 'front', 'of', 'a', 'white', 'fence', '<eos>', '<pad>', '<pad>', '<pad>'	'<sos>', 'a', 'white', 'white', 'is', 'running', 'in', 'a', 'grass', 'of', 'a', 'grass', 'grass', 'grass', 'grass', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>'
'<sos>', 'a', 'girl', 'in', 'karate', 'uniform', 'breaking', 'a', 'stick', 'with', 'a', 'front', 'kick', '<eos>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>'	'<sos>', 'a', 'girl', 'in', 'a', 'a', 'a', 'a', 'a', 'a', 'a', 'a', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>'
'<sos>', 'five', 'people', 'wearing', 'winter', 'jackets', 'and', 'helmets', 'stand', 'in', 'the', 'snow', 'with', '<unk>', 'in', 'the', 'background', '<eos>', '<pad>', '<pad>'	'<sos>', 'five', 'people', 'in', 'in', 'and', 'and', 'and', 'and', 'and', 'and', 'in', 'in', 'in', 'background', 'background', 'background', '<eos>', '<eos>'
'<sos>', 'people', 'are', 'fixing', 'the', 'roof', 'of', 'a', 'house', '<eos>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>'	'<sos>', 'people', 'are', 'the', 'the', 'the', 'a', 'a', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>'
'<sos>', 'a', 'group', 'of', 'people', 'standing', 'in', 'front', 'of', 'an', 'igloo', '<eos>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>'	'<sos>', 'a', 'group', 'of', 'people', 'standing', 'in', 'front', 'of', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>'
'<sos>', 'a', 'guy', 'works', 'on', 'a', 'building', '<eos>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>'	'<sos>', 'a', 'guy', 'working', 'on', 'a', 'a', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>'
'<sos>', 'a', 'man', 'in', 'a', 'vest', 'is', 'sitting', 'in', 'a', 'chair', 'and', 'holding', 'magazines', '<eos>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>'	'<sos>', 'a', 'man', 'in', 'a', 'yellow', 'jacket', 'is', 'sitting', 'a', 'a', 'chair', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>'
'<sos>', 'a', 'mother', 'and', 'her', 'young', 'song', 'enjoying', 'a', 'beautiful', 'day', 'outside', '<eos>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>'	'<sos>', 'a', 'mother', 'and', 'her', 'son', 'enjoying', 'enjoying', 'a', 'a', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<eos>', '<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.

Quantitatively speaking, the Full Transformer had better results. Better validation loss (1.6925 vs 3.1136), and better validation perplexity (5.4331 vs 22.5030).

Qualitatively speaking, neither architecture produced perfect translations. To achieve that, more data is likely necessary. However, the Full Transformer produced English sentences that are grammatically correct. Meanwhile, the encoder-only Transformer produced sentences that are grammatically incorrect, contain repeated words, etc.

This can be explained by the difference in how the two models generate the output sentence. The encoder-only model tries to map the input German sentence directly to the English tokens all at once, without considering the sequence of English words it has already produced. As a result, it does not have a way to enforce proper grammar or maintain consistency across time steps. This often leads to repeated words, incorrect phrasing, and broken sentence structure.In contrast, the Full Transformer contains both an encoder and a decoder. The encoder first reads and encodes the entire German sentence into a contextual representation. Then, the decoder generates the English translation one token at a time. At each step, the decoder attends to two things: (1) the encoded source sentence, and (2) the previously generated English words. This autoregressive process allows the model to build grammatically coherent sentences, because each new token is generated in the context of what has already been written.For this reason, the Full Transformer is better aligned with how translation naturally works, which explains why it produces more fluent outputs and achieves lower perplexity.

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.

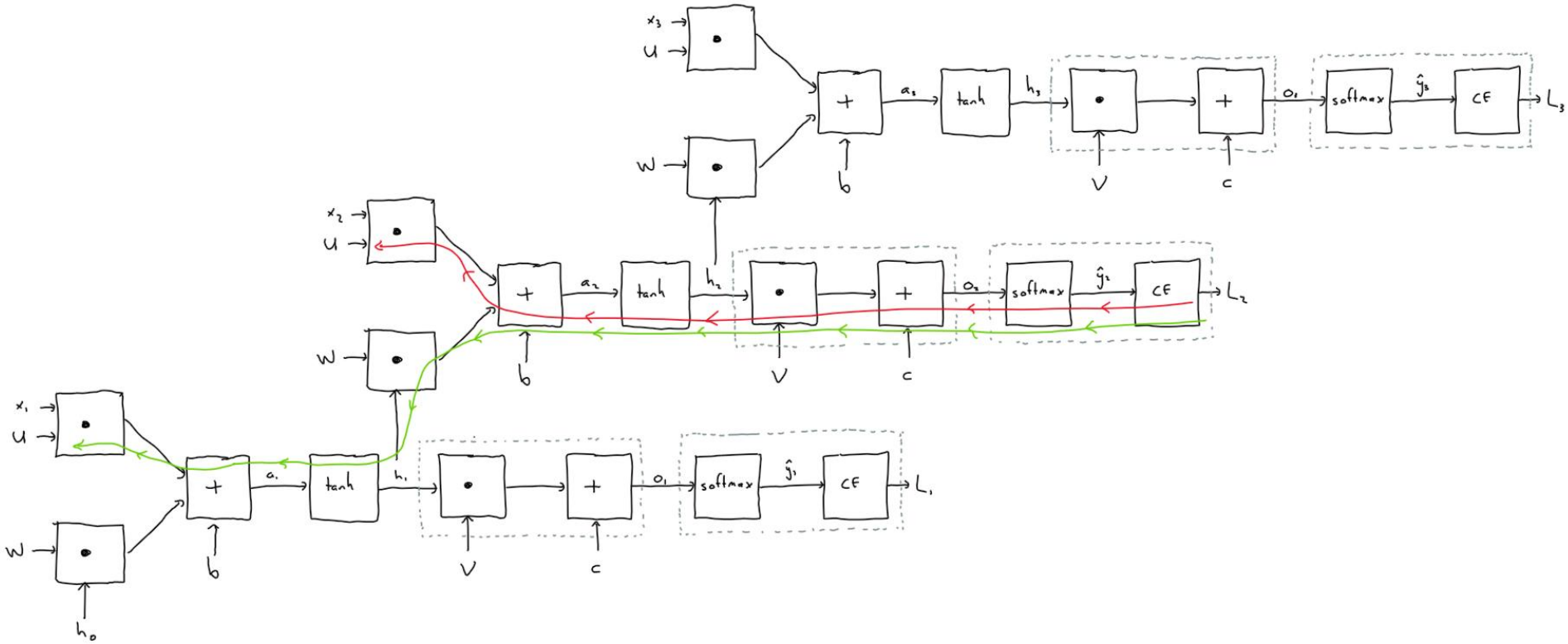
Quantitatively speaking, the transformer had better results. Better validation loss (1.6925 vs 3.1981), and better validation perplexity (5.4331 vs 24.4853).

Qualitatively speaking, neither architecture produced perfect translations. To achieve that, more data is likely necessary. However, the Transformer produced English sentences that are grammatically correct. Meanwhile, the Seq2Seq model produced sentences that are grammatically incorrect, contain repeated words, etc.

I believe the Transformer performs better because its self-attention mechanism allows every token to directly attend to all other tokens in the sequence. In contrast, the Seq2Seq model (even with attention) still relies on a recurrent hidden state that must carry information step-by-step through time, which makes long-range dependencies harder to capture. Since German and English often differ significantly in word order, being able to model global relationships at once gives the Transformer a clear advantage.

The Transformer also stacks multiple encoder and decoder layers, with multi-head attention enabling different heads to focus on different linguistic patterns (such as subject-verb agreement or phrase structure). This leads to richer representations and more coherent output. Meanwhile, Seq2Seq decoding is more prone to error accumulation—once an incorrect token is generated early on, later predictions often degrade, resulting in repetition or broken grammar. As a result, the Transformer not only achieves lower perplexity but also produces translations that are more fluent and grammatically consistent.

Theory question



$$\frac{\partial L_2}{\partial U} = \left(\frac{\partial L_2}{\partial U}\right)_2 + \left(\frac{\partial L_2}{\partial U}\right)_1$$

$$= \frac{\partial L_2}{\partial o_2}\frac{\partial o_2}{\partial h_2}\frac{\partial h_2}{\partial a_2}\frac{\partial a_2}{\partial U} + \frac{\partial L_2}{\partial a_2}\frac{\partial a_2}{\partial h_2}\frac{\partial h_2}{\partial a_2}\frac{\partial h_1}{\partial a_1}\frac{\partial a_1}{\partial U}$$

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

Language Models are Few-Shot Learners

What is the main contribution of this paper? In other words, briefly summarize its key insights. What are some strengths and weaknesses of this paper? What is your personal takeaway from this paper? This could be expressed either in terms of your perceived novelty of this paper compared to others you've read in the field, potential future directions of research in the area that the authors haven't addressed, or anything else that struck you as being noteworthy.

As someone who has been frequently using LLMs over the last few years, and as someone who has been recently introduced to machine learning, I have found myself asking the following question:

All of these model types that I am learning about seem to be trained to achieve a very specific task. Yet, the LLMs that I use seem to be capable at a wide range of tasks. What am I missing?

This paper answered this question for me. In 2020, these researchers shared the novel characteristic of LLMs, that they seem to be very competent at a wide range of tasks without being trained (or fine tuned) to perform well at that specific task.

Months before the release of this paper, the famous "scaling laws" paper was released by OpenAI. Another key contribution of this paper, is that it pushes the scaling laws further, and confirms that performance continues to scale predictably with compute, parameters, and training corpus size.

Other areas that were particularly notable to me include the discussion of the engineering challenges that they had in training GPT 3. Studying ML, it can be easy to think that architecture and hyperparameter tuning is the full scope of training a model. But at the scale of GPT 3, serious engineering constraints come into play. I enjoyed the mention of "model parallelism within each matrix multiply and model parallelism across the layers of the network". I did further research to gain a stronger intuition of what they had to do to manage the fact that GPT 3's activations and gradients did not fit on a single GPU.

I think another strength of this paper was highlighting its own weak points. One that stood out to me, was the admission that with such a large training corpus, it is likely that there was test contamination. Of course, I also appreciated the discussion on steps taken to mitigate that.

It is hard to talk about "potential future directions of research", since this type of model has been the spotlight of much attention for the last several years, and I know many of the future directions of research. One that I am particularly interested in, is the idea that chain of thought can provide such a boost in performance. The idea of a model checking its own work and iterating on it is fascinating to me.

How might we approach the technical limitations (e.g., changes in architecture, other context/data, optimization, etc.) mentioned in sections 5/6?

One of the limitations highlighted in section 5 is the lack of bidirectionality. The fact that GPT 3 only predicts tokens left to right, hinders it at performing tasks that would benefit from comparison. An interesting idea, which of course has been pursued, is chain of thought! Let the model read its own output, potentially spotting mistakes, and correcting them. All this “thinking” can happen before an answer is returned to the user.

Another challenge highlighted by the paper is poor sample efficiency. GPT 3 sees more data than any human does in a lifetime. Yet, it still underperforms compared to humans at many tasks. My intuition tells me that we may benefit from further studying how the human brain works, and potentially drawing ideas from that domain to develop more efficient architectures. Another idea that comes to mind, is somehow leveraging reinforcement learning. We know that humans don’t just learn from example, but also from experience. Perhaps this could help language models improve their sample efficiency.

Another concern mentioned in the paper is the limitation of the context window size. Something that comes to mind is the vanilla RNN, which can process arbitrarily long sequences of data, and always compress that into a fixed size vector. Of course, this has the risk of data loss, but it may be an idea that we can adapt in order to empower GPT models with infinite context window length.

What are the social implications of deploying these models for various uses (e.g., to generate image captions, answer questions as chatbots, etc.)?

This is the subject of much debate today. It is interesting to see that five years ago, the first researchers to develop an LLM, had enough foresight to mention some of the pressing challenges that we see in 2025.

One big challenge is AI chatbot dependency in humans. Especially children. These models are so capable that we are starting to see people's writing, coding, and general analysis skills atrophying.

Another concern is how easy it is to generate – on a massive scale – content that is hard to distinguish from content generated by humans. Text information is bad enough (fake articles, research studies, etc.). But it gets even worse when you realize that models that descended from GPT3 can also generate images, sound, and video that is hyper-realistic. We are already in a world where many of our interactions with reality happen through a screen. What happens when that reality can be easily forged?

Another big concern is the fact that language models have no sense of fairness or ethics. They simply model what is found in their training corpus. Since the training corpus for this scale of model is most of humanity's documented content, and since that documented content is littered with human bias, the result is that the models themselves carry those biases. As a quick test, I just asked ChatGPT (GPT 5) to "fill in the blank". I demonstrated that it assumes nurses are female and engineers are male. There are a wide range of techniques out there to mitigate these issues, ranging from modifyifying training data, to modifying already trained models. However, there is currently no perfect method for eliminating these potentially harmful biases.