Transformer model for translation and question-answering

Camilla Dybdal

March 4, 2022

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Contents

1 Introduction **1**

2 The datasets **2**
2.1 Preparing the datasets 2

3 Transformer model **4**

4 Training **6**

5 Results **8**
5.1 The question answering chatbot 8
5.2 The Russian to English translator 9

6 Improvements **12**
6.1 The question answering chatbot 12
6.2 The Russian to English translator 12

7 Conclusion **13**

References **14**

1 Introduction

When I started working om this project, I had a clear goal to create a witty deep-learning chatbot, using the Transformer model proposed in the paper "Attention is all you need" [1], which is known as state of the art within Natural language Generation (NLG). This however, turned out to be difficult as the chatbot showed very poor results, and I then proceeded to create a translator between Russian and English, using the same model. In this paper I will discuss the datasets I've used, the structure of the model, the training on both datasets and finally a discussion of the results. I'll especially focus on why the chatbot got such poor results compared to the translator, as this is the most important takeaway from my research.

Prior to this project, natural language processing was completely new for me. My initial idea was to use sequence to sequence RNN structure for the model. However, after discovering that this architecture is somewhat outdated, I decided to implement the Transformer architecture instead. The Transformer handles input by stacking self-attention layers, resulting data being processed in parallel.

My approach to implementing the chatbot and Russian-translator has been to follow a tutorial by Tensorflow, "Transformer model for language understanding" [6], where they teach you to create a translator from Portugese to English. I first created the translator in the tutorial, and then made my changes to the model to accustom it to a dataset instead consisting of questions and answers. To have some fun in the process, I've chosen a dataset consisting of jokes on the question-answer form. After training the chatbot, I proceeded to training the model using a Russian to English dataset.

Prior to the transformer model being proposed, the dominant sequence to sequence models within NLP were based on convolutional neural networks or RNNs. The transformer on the other hand, is based purely on attention mechanisms instead. In the paper "Attention is all you need" [1], the transformer model is being used for translating languages. This is also what it has been used for in the tensorflow-turtorial that I have followed in this paper. For this reason it was very exciting to try the model on a very different problem: question answering. Google have built a question-answering model based on the Transformer model (BERT), but with a little different angle. The google-model is trained on finding the answers to a question in a text. My problem differs from that, as it aims at teaching the model to form sentences solely based on the question.

When translating text, you may find a reasonable variety of outputs that are deemed correct according to the input. However, for question answering, the scope of valid answers are in a completely different category. For a single question, there can be an infinite amount of possible answers, resulting in a very low correlation between input and output.

2 The datasets

The dataset used for the chatbot is obtained from Kaggle [3], and contains 38269 jokes, or riddles if you want, on the question-answer format. This dataset is small for the task, which could lead to a model more prone to overfitting. In comparison, the dataset in the tensorflow turtorial which I'm using as my guide, contains 50000 training examples, and obtained an accuracy of about 0.7 after training. Since translation problems generally have larger linguistic semantics than question-answering, having this little data for teaching the model language is prone to be a bit tricky.

Since the dataset originates from a forum, it contains everything from one-worded puns to full sentences as answers. As the goal is to create a chatbot that provides witty responses to questions, having a broad examples of witty answers is for sure good. This can also help prevent the model from overfitting, as the questions and answers doesn't conform to a particular format or category. However, after doing some manual inspection of the dataset, I found that some questions repeat themselves with small variations. Another issue I found was that most the answers are very short, in comparison to the input questions. This could result in a model that is very hard to train, as there is little correlation between the input and the output.

The dataset used for the Russian to English translator is obtained from Tensorflow datasets [5], and contains 208,106 training samples, 5,476 test samples and 4,805 validation samples. This is a relatively large dataset, which is why I chose Russian for the translation task. An interesting aspect of translating from Russian, is that the Russian language uses the Cyrillic alphabet, as seen in fig. 2.2, and the English language uses Latin, resulting in very different tokens for the two languages.

2.1 Preparing the datasets

Downloaded from Kaggle, the question answering dataset needed a bit of preprocessing before it could be used in the model. The first thing to do was to split the data into training and test examples, as well as splitting questions from answers. The python script for doing this can be found in the file called *dataprocessing.py* in the project on github [2]. This wasn't needed for the Russian dataset, as it was already split up and ready to use.

Since you can't train the model directly on text, the data needed to be converted into numerical values. One very popular method for this is using BERT tokenizer, developed by Google. What is interesting about this tokenizer is that it doesn't just convert each word or each letter into tokens, but instead splits uncommon words into "subwords", so it can fall back on word pieces and individual characters for uncommon words in the dataset. This is shown in fig. 2.1, where the word "Mycheexarphlexin", which obviously isn't a real english word, is split up into my, che, ex, ar, ph,

le, x and in. This tokenizer also splits punctiation, lowercases and unicode-normalizes the data before tokenizing. I made tokenizers for both datasets by following a Tensorflow-tutorial [4].

```python
for answers in train_answers.batch(3).take(1):
  for a in answers:
    print(a.numpy())
```

```
b'He nearly drown in his own tea pee.'
b'Mycheexarphlexin'
b'Matt'
```

```python
txt_tokens = tf.gather(questions_vocab, token_batch)
tf.strings.reduce_join(txt_tokens, separator=' ', axis=-1)
```

```
<tf.Tensor: shape=(3,), dtype=string, numpy=
array([b'he near ##ly drown in his own tea pee .',
       b'my ##che ##ex ##ar ##ph ##le ##x ##in', b'm ##at ##t'],
      dtype=object)>
```

Figure (2.1) Examples from the question answering dataset before and after tokenization

```python
print(ru_vocab[:10])
print(ru_vocab[100:110])
print(ru_vocab[1000:1010])
print(ru_vocab[-10:])
```

✓ 0.7s Python

```
['[PAD]', '[UNK]', '[START]', '[END]', '!', '#', '$', '%', '&', "'"]
['ш', 'щ', 'ъ', 'ы', 'ь', 'э', 'ю', 'я', 'і', 'ґ']
['трудно', 'хотела', 'далеко', 'качестве', 'мою', '##З', '##де', '##ила', 'планеты', 'большие']
['##’', '##“', '##”', '##„', '##•', '##´', '##/', '##∇', '##♪', '##♫']
```

Figure (2.2) Examples from the Russian vocabulary

3 Transformer model

The Transformer model consists of input embedding and positional encoding, an encoder block, decoder block and a linear layer with a proceeding softmax layer at the end, as can be seen in fig. 3.1. The input embedding and positional encoding gives us word-information that has positional context.

The encoder layer consists of a multi-head attention layer as well as a feed-forward network. The attention layer outputs attention vectors for every single token, and ultimately tells the model which part of the input it should focus on. The feed-forward network is applied to all attention vectors, and transforms the vectors into digestible parts for the next blocks. Each of the layers has a residual connection around it which is then followed by a layer normalization.

The decoder layer consists of a masked multi-head attention layer (with a look-ahead mask), another multi-head attention layer and lastly a feed-forward network. Also here does the layers have a residual connection around them which is then followed by a layer normalization. What is most interesting here is the input to the mutli-head attention layer, which received the encoder outputs (K,V) as inputs. Q is received from the masked mutli-head attention layer. This is where self-attention comes in, as the decoder predicts the next token by using the encoder output while self-attending to its own output. Both the encoder and decoder have dropout layers to prevent overfitting. All the multi-headed attention layers also have a padding mask, so that the padding added to the sentence isn't being treated as input.

This brings us to the most important and interesting mechanism in the Transformer model, namely attention. This mechanism has the effect of giving more attention to important parts of the data. Multi-head simply means the attention mechanism is ran through several times in parallel. Within this layer scaled dot product attention, given in eq. (3.1), is used as the attention function. The inputs to this function consists of Q, query, K, keys and V, values of dimension d_k. Softmax is applied to get the weights on the values.

$$Attention(Q, K, V) = softmax_k(\frac{QK^T}{\sqrt{d_k}}) \tag{3.1}$$

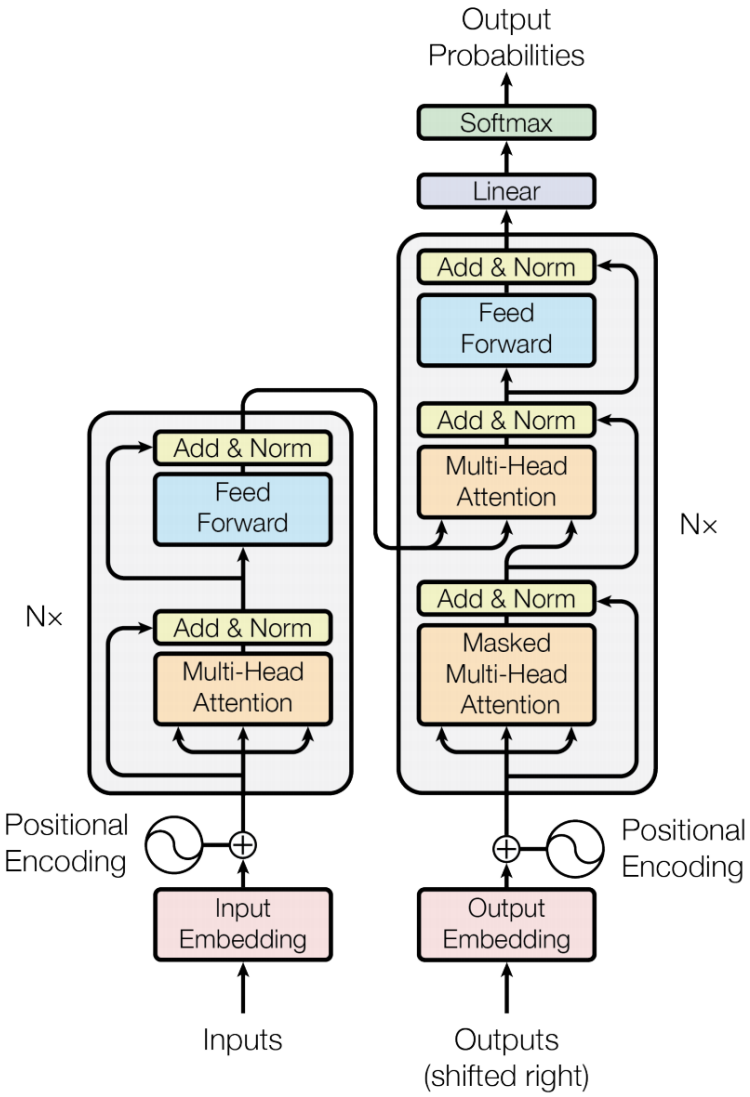

Figure (3.1) The transformer model architecture [1]

4 Training

In the paper "Attention is all you need" [**1**], the hyperparameters were set as shown in table 4.1, resulting in a pretty deep network which takes a very long time to train. They spent 12 hours training the base models, and 3.5 days for their big models. This was using relatively decent GPUs, which I'm not equipped with myself. They were also using much larger datasets, containing 4.5 million and 36 million training pairs. I used the hyperparamaters shown in table 4.1 for training the chatbot on the question anwering dataset, as this datset is relatively small. This took me about 7 hours, for 20 epochs. For training the Russian to English translator I used the hyperparameters given in table 4.2, training over 12 epochs, which took about 2.5 days.

Table (4.1) Hyperparameters used in the paper "Attention is all you need"

Hyperparameters[**1**]	
num_layers	6
d_model	512
d_{ff}	2048
num_heads	8
$dropoutrate$	0.1

Table (4.2) Hyperparameters used when training Russian-English translator

Hyperparameters	
num_layers	4
d_model	128
d_{ff}	512
num_heads	8
$dropoutrate$	0.1

Both the paper and tutorial was using the Adam optimizer, according to the formula in the paper [1]. This optimizer has a linearly increasing learning rate during the warm-up training steps, which is set to 4000, and then decreases it proportionally to the inverse square root of the step number, which can be seen in fig. 4.1. The loss is calculated using the Keras function for cross-entropy. Since the target sequence is padded, the padding mask was added to the calculations of loss and accuracy as well. The dropout rate is added to prevent overfitting.

The Transformer is an auto-regressive model, meaning the predictions depend linearly on its own previous values, and uses its output so far to decide what to do next. Teacher forcing is used when training, allowing us to pass the true output to the next time step, regardless of what the model predicted at the current time step.

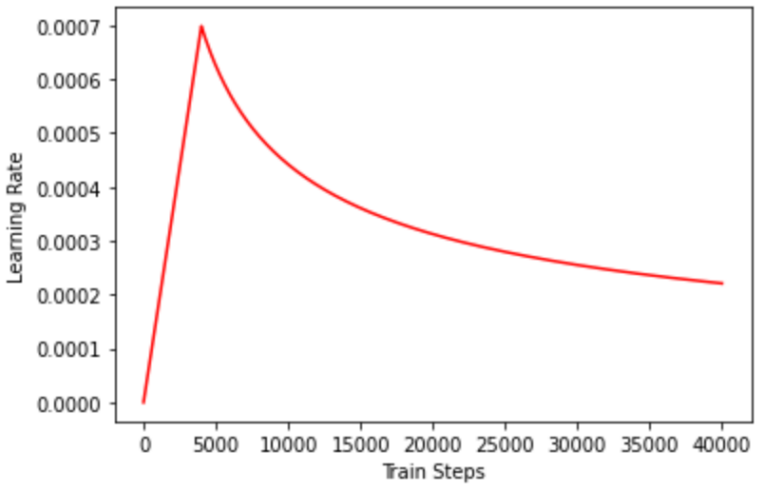

Figure (4.1) The learning rate from the Adam optimizer

5 Results

Evaluation of results from an NLP model can be tricky, especially in the case for the question answering model. This is because for one single input, there can be many valid and also good outputs, differing from the ground truth by far. Thus for evaluating the chatbot results I will qualitatively inspect the output. For the Russian to English translator I will use the BLEU-metrics when evaluating the results.

5.1 The question answering chatbot

As can be seen in fig. 5.1, the chatbots' prediction to the questions make absolutely no sense. There are many odd tokens in the answer, and no matter what I ask it, it always has "golden" in the answer. This is odd, as "golden" is mentioned only about 30 times in the dataset. The word "deadress" also reappears in many of the outputs from the chatbot, which is even odder as this is a completely made up word from the chatbot. I tried to train for even more epochs to see if this would result in higher accuracy, but it instead resulted in even worse outputs. This implies that the model was already overfitted, and training for more epochs only increased this effect.

```
sentence = "Whats got wrinkles and hangs out your undies?"
ground_truth = "Your granny"

translated_text, translated_tokens, attention_weights = chatbot(tf.constant(sentence))

print_interaction(sentence, translated_text, ground_truth)
```

✓ 2.1s Python

```
Input:         : Whats got wrinkles and hangs out your undies?
Prediction     : golden 清 ' old them π ι a like wing nuted n deer ♪ .
Ground truth   : Your granny
```

```
sentence = "Want to hear a word I just made up?"
ground_truth = "Plagerism"

translated_text, translated_tokens, attention_weights = chatbot(tf.constant(sentence))

print_interaction(sentence, translated_text, ground_truth)
```

✓ 4.8s Python

```
Input:         : Want to hear a word I just made up?
Prediction     : _ : golden ╦ “ _ deadress .
Ground truth   : Plagerism
```

Figure (5.1) Example of the chatbots' responses to questions in the training set.

The attention plots for all attention heads are shown in fig. 5.2, where a darker color implies less attention and a lighter one implying more. The plots seems to be random, and it is very hard to see a pattern between the input and the prediction. If looking closely, we can see that there is a connection between "." from the prediction and the "[START]" and "[END]" tokens from the input, which is most apparent in head 2, 5 and 8. Other than this, it is hard to see any correlation.

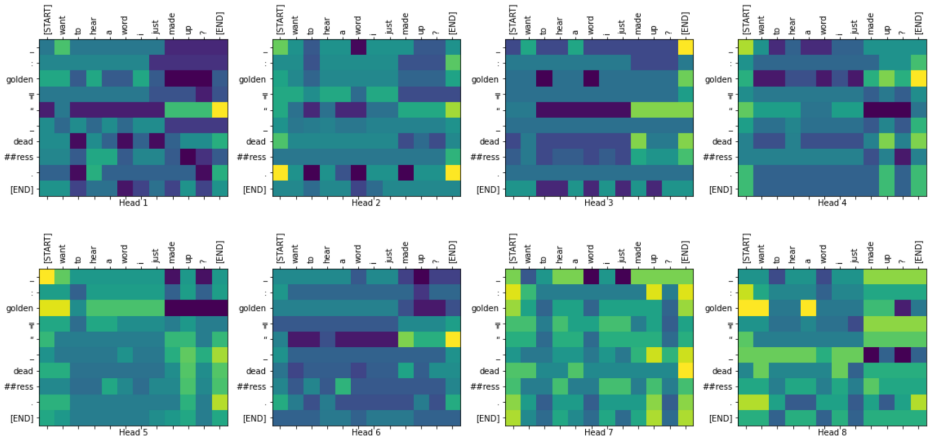

Figure (5.2) The attention plot from the chatbot for all attention heads in layer 4.

It is worth mentioning that the model achieved an accuracy of 0.4441 and a loss of 2.5731 after training for 50 epochs. Even though these aren't impressive results, they do imply a better expected behavior than the one observed when testing the model. I have tested the model with over 15 different inputs, from both the test and training data, and it's all the same nonsense you see in fig. 5.1.

5.2 The Russian to English translator

The Russian to English translator did much better than the chatbot. As previously mentioned, evaluation of text outputs in NLP-problems is a complex task. I've used the BLEU (BiLingual Evaluation Understudy) metrics for evaluating of my translator, even though I believe its a rather harsh way of evaluating success. BLEU compares each word in the prediction with the ground truth, or optionally a bunch of references for the ground truth. Since my dataset didn't include variations of the same sentence, I've only used a single reference as the ground truth for each test sample.An identical sentence receives a BLEU score of 1, and one with no identical words gets 0. This means that even if the translation is good, or at least a valid translation of the input, it will still get a low BLEU score if the words aren't identical.

The translator obtained a final BLEU score of 14.8 (0.148), which is considered poor compared to the score of 28.4 obtained in the English to German translator from the paper "Attention is all you need" [1]. The translator is far from good, however, I believe the BLEU score doesn't reflect very well the actual performance, and that a qualitative visual inspection might tell us more in this case. As you can see in fig. 5.3, the translator successfully translates the first sentence, resulting in a BLEU value of 1.0. In the other two examples, we see that the translator catches the general

meaning, but fails to translate the sentences correctly. "Hard things" and "challenges" have similar meanings, however, they're not the same, which justifies the BLEU score of 0.54. However, in the last example we get a BLEU score of basically 0, even though the translator gets the general meaning of the sentence.

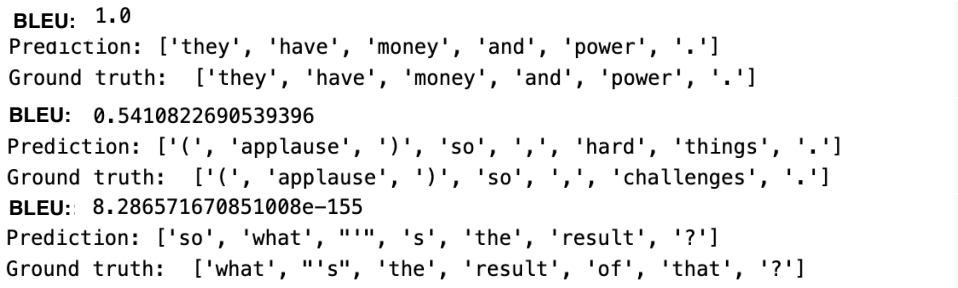

Figure (5.3) Three examples of BLEU scores and their corresponding sentences.

The attention plots for the Russian to English translator in fig. 5.4 portray a much clearer trend compared to the ones for the chatbot shown in fig. 5.2. By looking at the attention plots, we see that there is more attention along the diagonal. The plot is a little difficult to read, as it seems the Russian and English tokens are split differently. We also see that a lot of attention is put on the "[END]" token in the Russian sentence when predicting each word in the English translation. This is unwanted behavior, as the attention preferably would lie on the corresponding English token, and might contribute to the model being off. This trend could be a result of the English tokens being so split up compared to the Russian ones, resulting in the model not being able to find a good correlation other than the "[END]" token, which is always there.

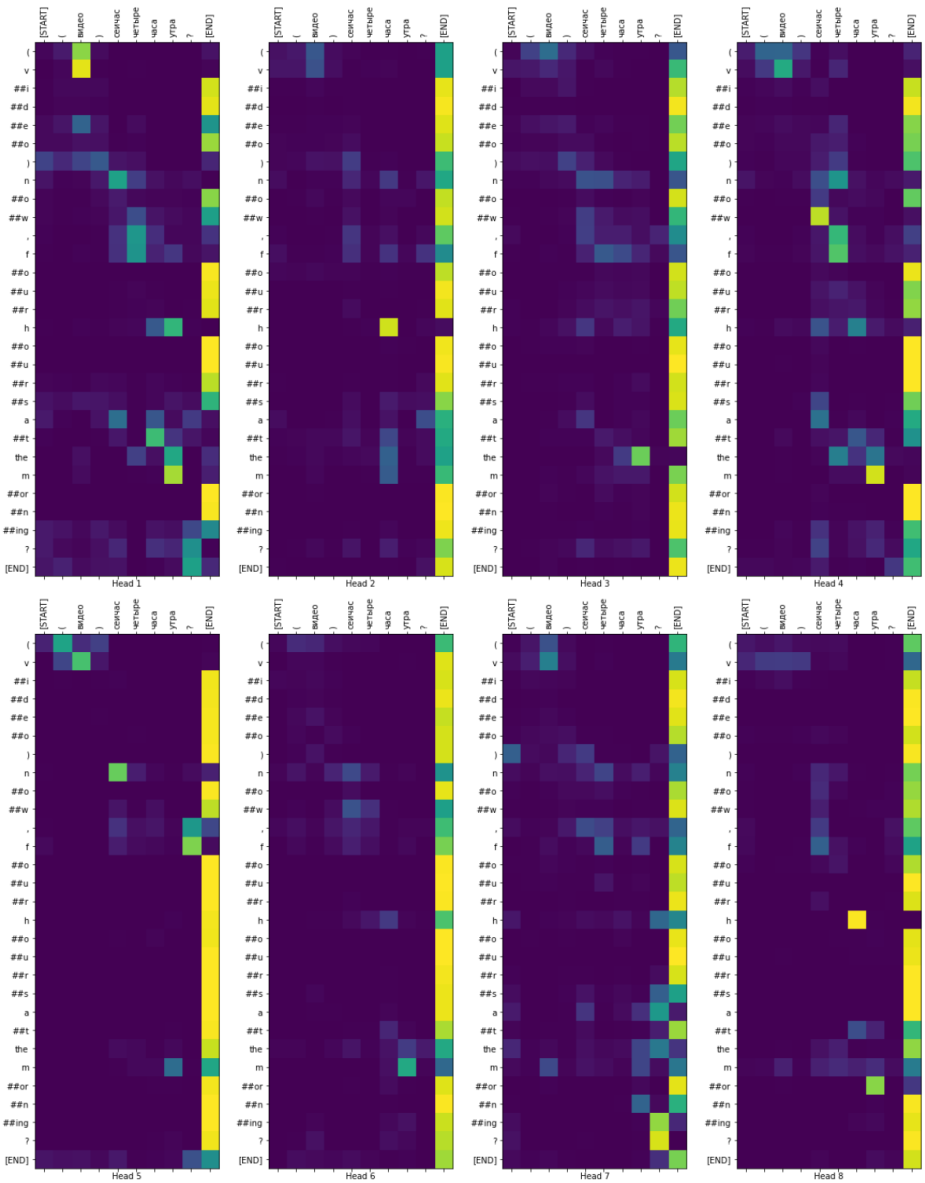

Figure (5.4) The attention plot from the Russian to English translator for all attention heads in layer 4.

6 Improvements

6.1 The question answering chatbot

As already established, the chatbot has potential for improvements (to say the least). The most probable reason for its poor results is the dataset not being suitable for the task. One reason for this is possibly the format of the questions and answers; they're riddles. The dataset contains answers sentences which vary from being one word long and not having any true meaning to being long sentences. For most humans, knowing the answer to a riddle you've never heard before is already next to impossible. For a model, which doesn't even understand English from before, this is probably next to impossible to learn from. There is little to no correlation between the questions and the answers, making it very hard for a model looking to build a general structure. One improvement area would be to find a different dataset, prefferably larger, and with better sentences and answers.

If the model already knew how to read and form meaningful sentences, it would have definitely been easier to teach it jokes. This brings me to my most important improvement area for the chatbot, which is to apply transfer learning. Using a pre-trained model and fine tuning it with the jokes dataset probably would have given decent results.

6.2 The Russian to English translator

An obvious area of improvement in this task is having more patience, or a better GPU to run the training on. The accuracy was still increasing, and the loss decreasing, when I ended training. Iterating over even more epochs would probably provide with better results than the ones obtained.

Another area of improvement would be to figure out a way to tokenize the two languages in a more similar manner, possibly not by using a sub-word tokenizer. As previously mentioned, the unwanted trend we saw in the attention plots in fig. 5.4 could have been a result from the English and Russian tokens being split up differently, which throws the network off when looking for a correlation.

7 Conclusion

The most important finding in this project was the difference in performance between the Russian to English translator and the question answering chatbot. Even though I wouldn't brag about the results observed from any of them, the translator worked while the chatbot didn't. The same Transformer was used for both the translator and chatbot, which clearly shows the importance in the choice of dataset. The dataset chosen for the chatbot had too little correlation between the questions and answers, making it impossible for the model to generalize. The Russian to English dataset had far more correlation, which makes sense as similar semantics between languages is typical .

References

[1] *Attention is all you need.* https://arxiv.org/abs/1706.03762. Accessed: 2022-25-02.

[2] *Final project github.* https://github.com/camilladybdal/transformer_project. Accessed: 2022-25-02.

[3] *Question-Answer Jokes.* https://www.kaggle.com/jiriroz/qa-jokes/version/2. Accessed: 2022-25-02.

[4] *Subword tokenizers.* https://www.tensorflow.org/text/guide/subwords_tokenizer. Accessed: 2022-25-02.

[5] *ted_hrlr_translate.* https://www.tensorflow.org/datasets/catalog/ted_hrlr_translate#ted_hrlr_translatetr_to_en. Accessed: 2022-25-02.

[6] *Transformer model for language understanding.* https://www.tensorflow.org/text/tutorials/transformer#text_tokenization_detokenization. Accessed: 2022-25-02.