CSE40201 - Natural Language Processing
Assignment 3
Neural Machine Translation

Student name: Nguyen Minh Duc
Student ID: 20202026

1 Implementing sequence-to-sequence model with attention

(a) In this first problem, we are asked to implement the encoder, decoder, training, and inference code. The solution is in `assn3_1_seq2seq_attention.ipynb`. In the process of implementing the model, I have derived the following hyperparameters

```
EMBEDDING_SIZE = 256            # Size of the embedding vector
ENCODER_HIDDEN_SIZE = 1024      # Size of the hidden state in the encoder
DECODER_HIDDEN_SIZE = 1024      # Size of the hidden state in the decoder
N_LSTM_LAYERS = 1               # The number of stacked LSTM layers
DROPOUT_P = 0                   # The dropout probability
```

I will use the hyperparameters mentioned above to train the first model in this part. The learning rate is kept at the default value, i.e., 5×10^{-4}. The training took about 10 minutes to finish as each epoch took 54 seconds. The following will contain translation results for Examples 6, 7, 29, and 500 in the validation set and attention visualization of them. Please refer to Table 1 for translation results, and Figures 1, 2, 3, and 4 for visualizations.

Example	Source Target Translation output
6	['ein', 'brauner', 'hund', 'rennt', 'dem', 'schwarzen', 'hund', 'hinterher', '.'] ['a', 'brown', 'dog', 'is', 'running', 'after', 'the', 'black', 'dog', '.'] ['a', 'brown', 'dog', 'runs', 'running', 'black', 'dog', '.', '<eos>']
7	['ein', 'kleiner', 'junge', 'mit', 'einem', 'giants-trikot', 'schwingt', 'einen', 'baseballschläger', 'in', 'richtung', 'eines', 'ankommenden', 'balls', '.'] ['a', 'young', 'boy', 'wearing', 'a', 'giants', 'jersey', 'swings', 'a', 'baseball', 'bat', 'at', 'an', 'incoming', 'pitch', '.'] ['a', 'little', 'boy', 'wearing', 'a', 'baseball', 'cap', 'swings', 'a', 'bat', 'at', 'a', 'soccer', 'ball', '.', '<eos>']
29	['ein', 'traktor', 'bewegt', 'erde', 'für', 'den', 'bau', 'einer', 'stützmauer', '.'] ['a', 'tractor', 'is', 'moving', 'dirt', 'to', 'help', 'build', 'up', 'a', 'retaining', 'wall', '.'] ['a', 'small', 'male', 'is', 'driving', 'up', 'a', 'a', 'in', 'a', 'backyard', '.', '<eos>']
500	['eine', 'frau', 'zieht', 'einem', 'kleinen', 'mädchen', 'einen', 'helm', 'an', '.'] ['a', 'woman', 'is', 'putting', 'a', 'helmet', 'on', 'a', 'small', 'girl', '.'] ['a', 'woman', 'is', 'a', 'little', 'girl', 'a', 'a', 'a', 'stroller', '.', '<eos>']

Table 1: Example outputs on validation set

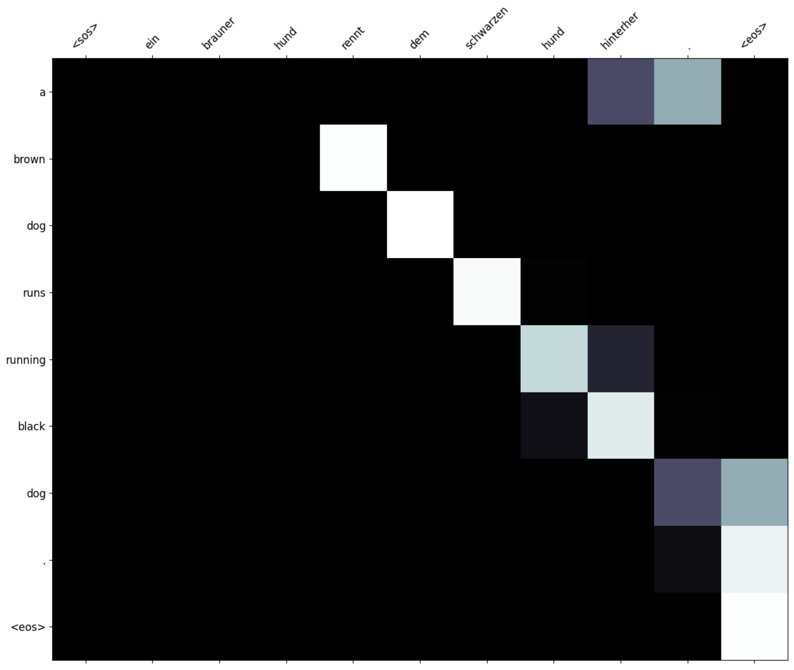

Figure 1: Attention heat map for Example 6 in validation set

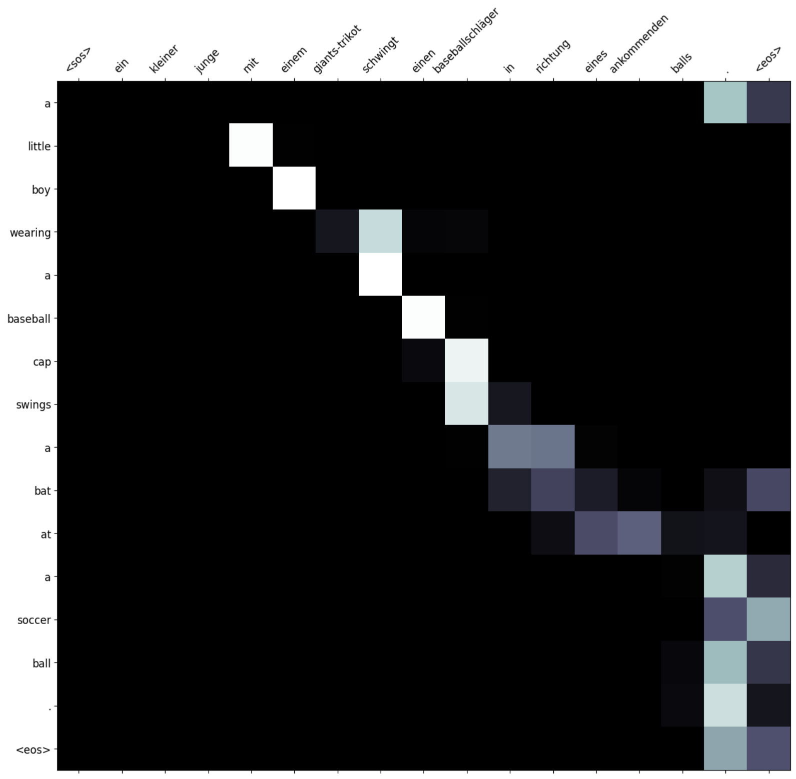

Figure 2: Attention heat map for Example 7 in validation set

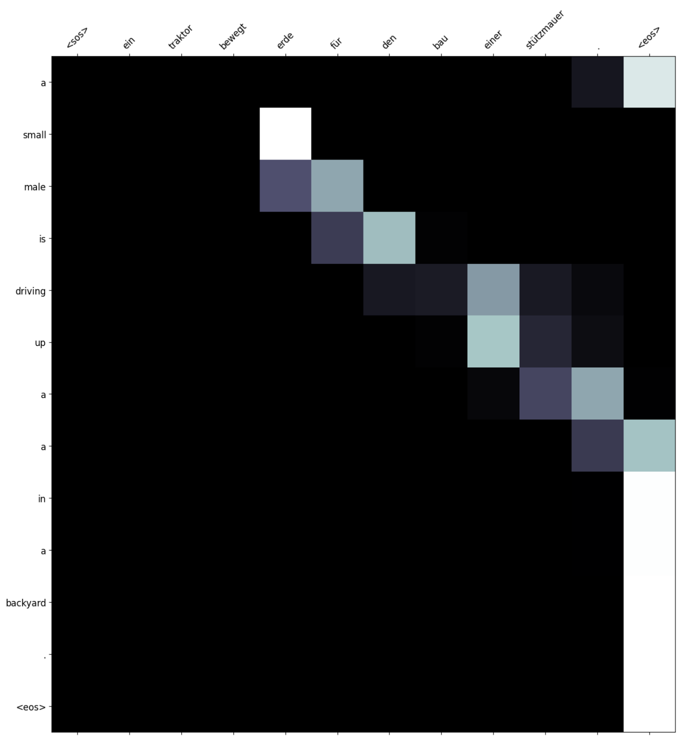

Figure 3: Attention heat map for Example 29 in validation set

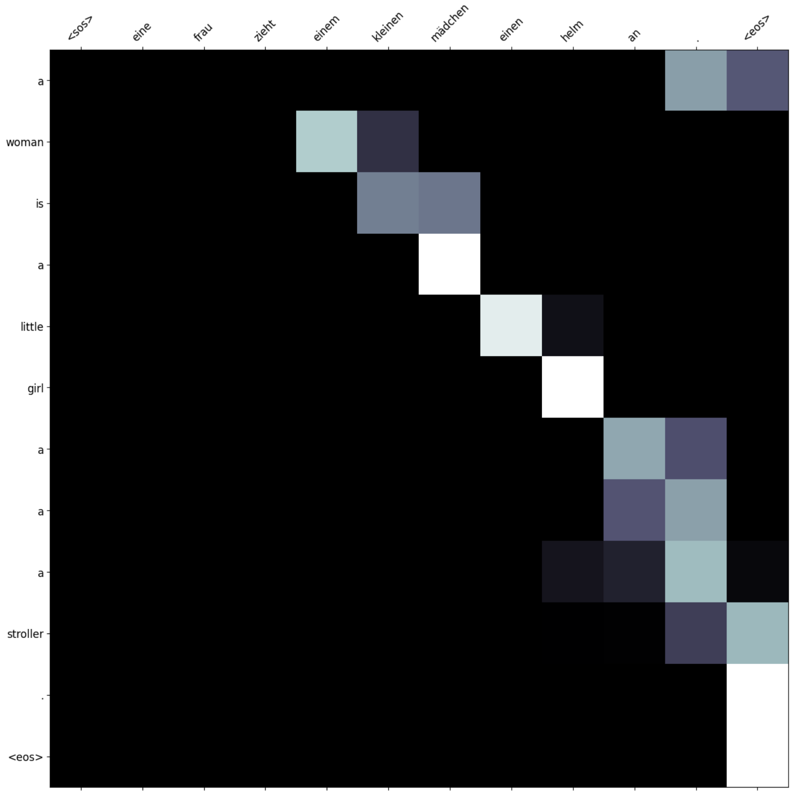

Figure 4: Attention heat map for Example 500 in validation set

The BLUE score for the test set is 32.30.

(b) In this second problem, we are asked to tune our model to get the best performance we can get. By trying a lot of different combinations of hyperparameters, I have achieved a BLUE score of 29.19 with the following hyperparameters:

```
EMBEDDING_SIZE = 1024
ENCODER_HIDDEN_SIZE = 512
DECODER_HIDDEN_SIZE = 512
N_LSTM_LAYERS = 1
DROPOUT_P = 0
```

Each epoch took 30 seconds, and the training finished after around 5 minutes on Google Colab. I notice that increasing the size of the embedding vector generally improves the model performance since the embedding layer can have a better representation of word vectors with a higher dimension. As for the hidden sizes, increasing them too much can lead to overfitting and slower training time. I also tried to stack more LSTM layers but it overfitted easily, so I decided to keep this as 1, hence, the dropout probability is 0. With all of that information, I find the above hyperparameter works well with our model.

(c) In this third problem, we are asked to implement the bidirectional version of LSTMs for the encoder. The implementation is quite straightforward as I only needed to add `bidirectional=True` to the `nn.LSTM` module. Another change was that I had to concatenate the forward and backward hidden state of the LSTM before feeding them into the decoder.

I used the same hyperparameter as part (b) except for the two hidden sizes where I reduce them to 256 (using the hidden size of 512 resulted in the size final hidden state being too big, hence, easier to overfit) and achieved a BLEU score of 32.30 after around 5 minutes of training, which is much better than the unidirectional LSTM implemented above. The bidirectional LSTM performs the learning in both directions, giving the current state information of both past and future tokens, hence, providing a better knowledge, and context representation of the current state.

The implementation can be found in `assn3_1_seq2seq_attention_bidirectional.ipynb`.

2 Implementing Transformer model

(a) In this section, we are asked to implement the Transformer architecture, in particular, the encoder, decoder, training, and inference code. The solution is in `assn3_2_transformer.ipynb`.

I used the default hyperparameters that were already in the code. The learning rate was also kept at the default value, i.e., 5×10^{-4}. The training only took about 2 minutes to finish as each epoch took 14 seconds. The following will contain translation results for Examples 6, 7, 29, and 500 in the validation set and attention visualization of them. Please refer to Table 2 for translation results, and Figures 5, 6, 7, and 8 for visualizations.

The BLUE score for the test set is 35.04, which is better than both LSTM models with a significantly faster training time.

Example	**Source** **Target** **Translation output**
6	['ein', 'brauner', 'hund', 'rennt', 'dem', 'schwarzen', 'hund', 'hinterher', '.'] ['a', 'brown', 'dog', 'is', 'running', 'after', 'the', 'black', 'dog', '.'] ['a', 'brown', 'dog', 'running', 'after', 'a', 'black', 'dog', '.', '<eos>']
7	['ein', 'kleiner', 'junge', 'mit', 'einem', 'giants-trikot', 'schwingt', 'einen', 'baseballschläger', 'in', 'richtung', 'eines', 'ankommenden', 'balls', '.'] ['a', 'young', 'boy', 'wearing', 'a', 'giants', 'jersey', 'swings', 'a', 'baseball', 'bat', 'at', 'an', 'incoming', 'pitch', '.'] ['a', 'young', 'boy', 'with', 'a', 'baseball', 'cap', 'swings', 'a', 'baseball', 'bat', 'towards', 'an', '<unk>', ',', '<unk>', ',', '<unk>', '.', '<eos>']
29	['ein', 'traktor', 'bewegt', 'erde', 'für', 'den', 'bau', 'einer', 'stützmauer', '.'] ['a', 'tractor', 'is', 'moving', 'dirt', 'to', 'help', 'build', 'up', 'a', 'retaining', 'wall', '.'] ['a', 'tractor', 'moving', 'dirt', 'for', 'dirt', '.', '<eos>']
500	['eine', 'frau', 'zieht', 'einem', 'kleinen', 'mädchen', 'einen', 'helm', 'an', '.'] ['a', 'woman', 'is', 'putting', 'a', 'helmet', 'on', 'a', 'small', 'girl', '.'] ['a', 'woman', 'is', 'pulling', 'a', 'little', 'girl', 'in', 'a', 'helmet', '.', '<eos>']

Table 2: Example outputs on validation set

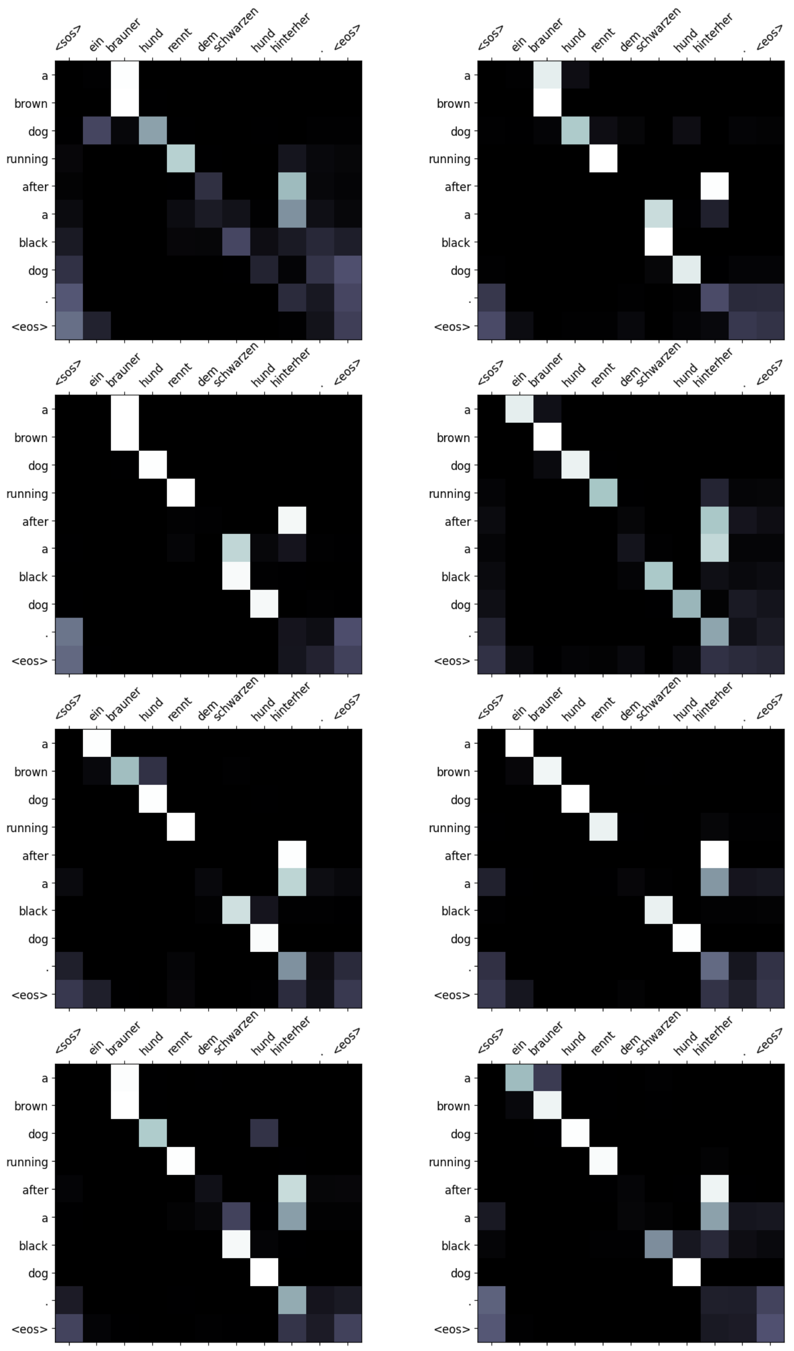

Figure 5: Attention heat map for Example 6 in validation set

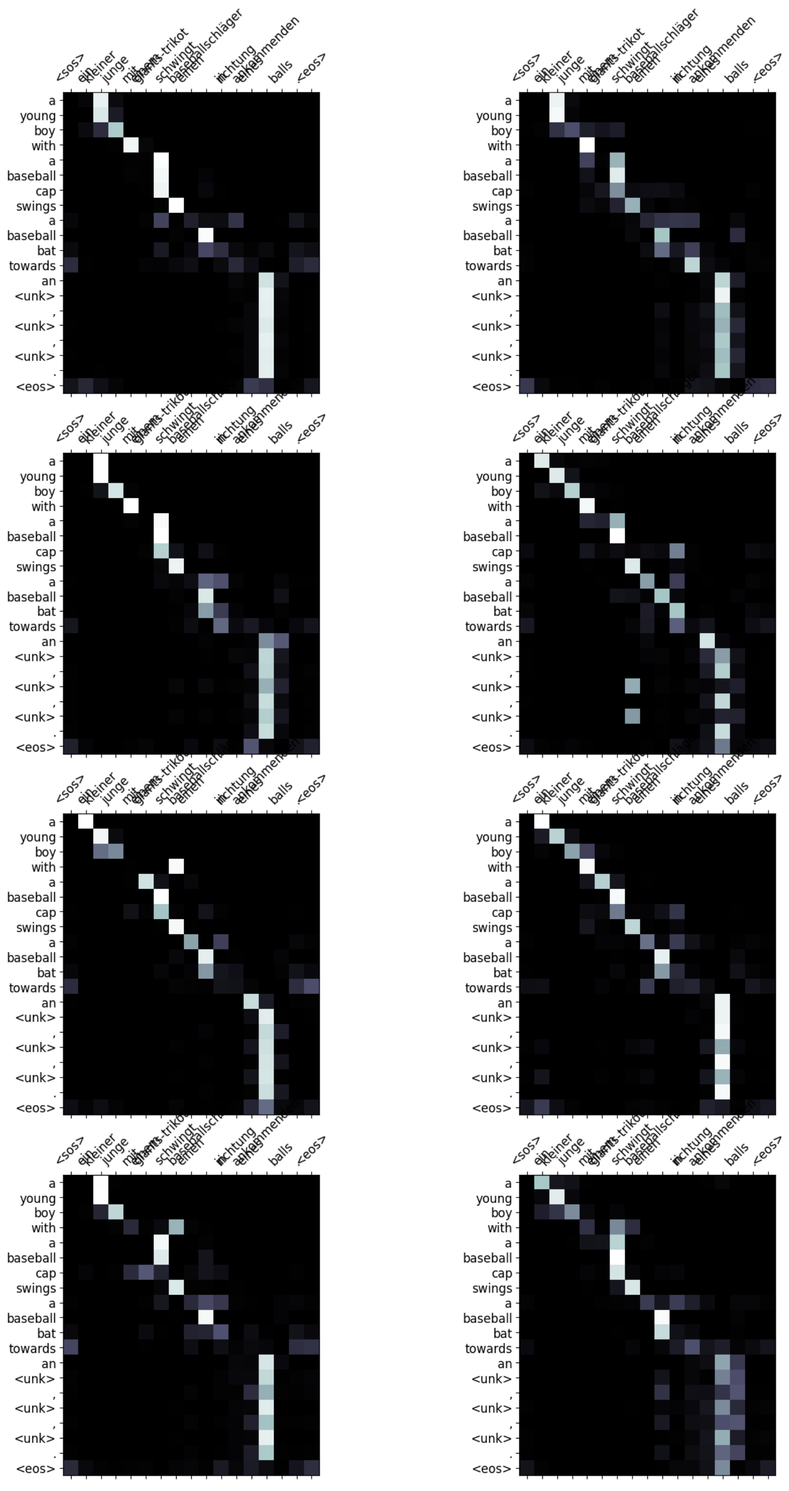

Figure 6: Attention heat map for Example 7 in validation set

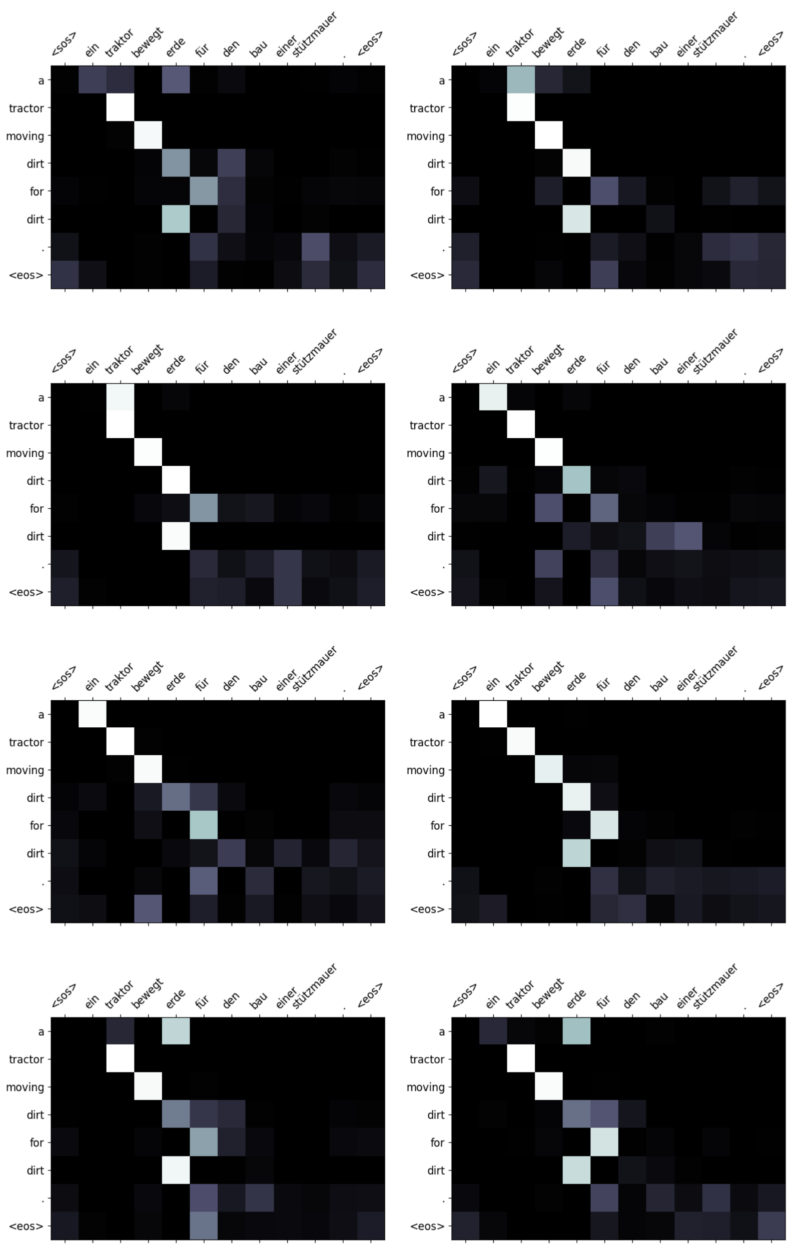

Figure 7: Attention heat map for Example 29 in validation set

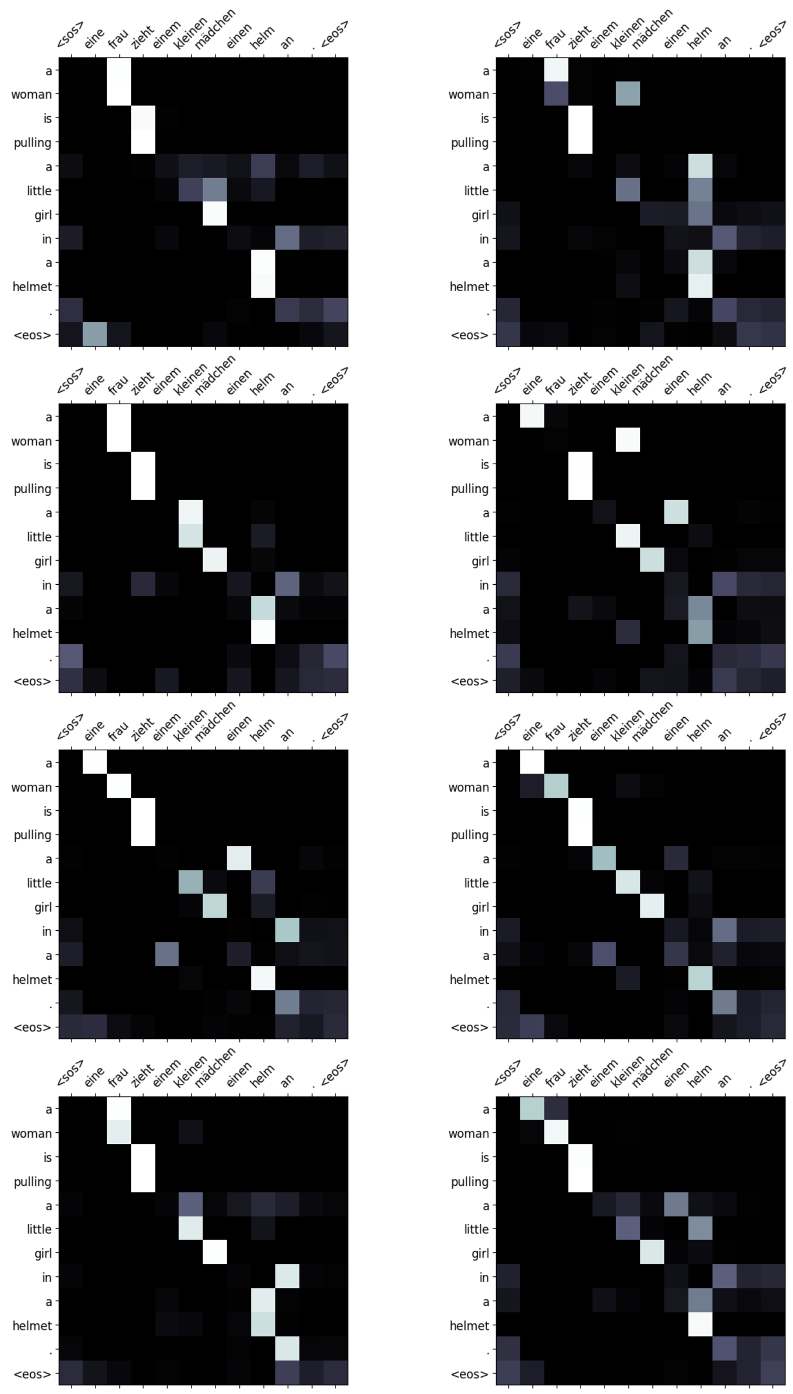

Figure 8: Attention heat map for Example 500 in validation set

(b) In this part, we are asked to tune our model to get the best possible performance. By switching the activation function in the `PositionwiseFeedforwardLayer` from `ReLU` to `GELU` following the BERT architecture [1], I am able to achieve a BLEU score of 36.37 on the test set with the same hyperparameter. `GELU` was used in other multiple well-known transformer-based models such as the GPT family [3] [4]. According to my findings, `GELU` is smoother, and more stable compared to `ReLU`. With its increased curvature and non-monotonicity, `GELU` can approximate complicated functions more easily [2].

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805, 2019.

[2] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv: Learning*, 2016.

[3] Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018.

[4] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.