〈Assignment 2〉

2022547Z 박상준

1-(a).

Translation result :

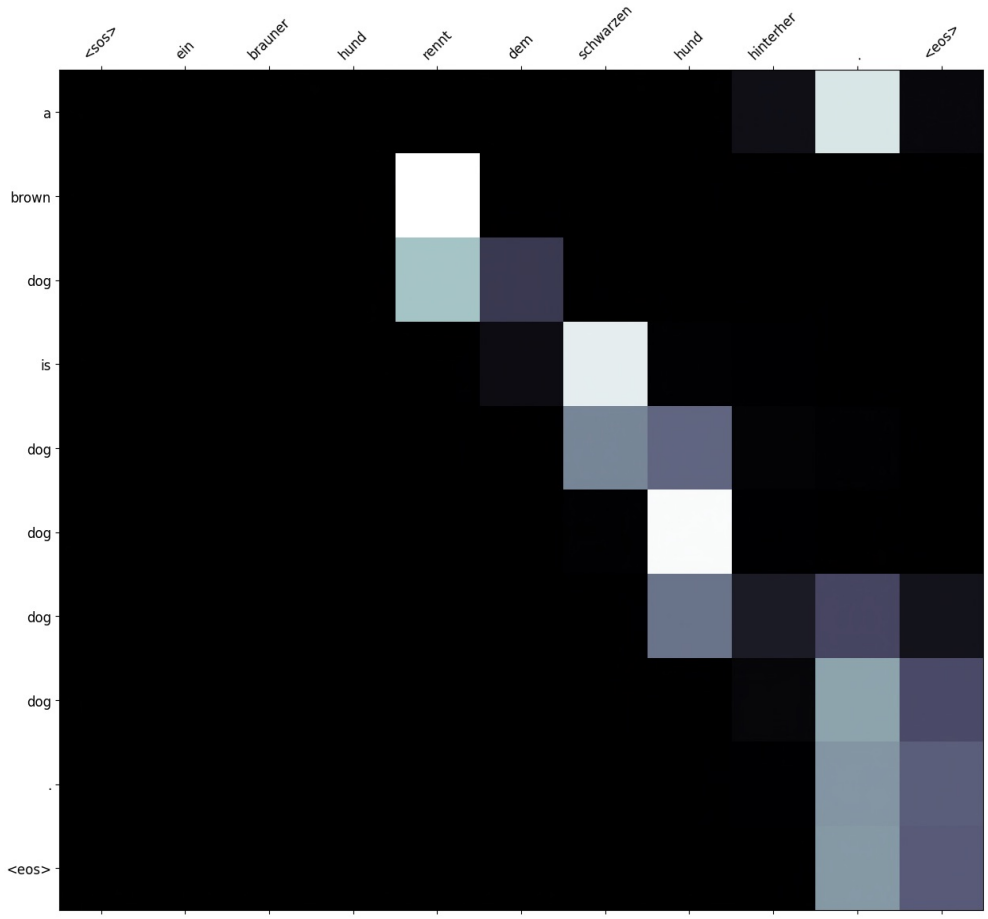
```
example_idx = 6

src = vars(valid_data.examples[example_idx])['src']
trg = vars(valid_data.examples[example_idx])['trg']

print(f'src = {src}')
print(f'trg = {trg}')
```

```
src = ['ein', 'brauner', 'hund', 'rennt', 'dem', 'schwarzen', 'hund', 'hinterher', '.']
trg = ['a', 'brown', 'dog', 'is', 'running', 'after', 'the', 'black', 'dog', '.']
```

```
translation, attention = translate_sentence(src, SRC, TRG, model, device)

print(f'predicted trg = {translation}')
```

```
predicted trg = ['a', 'brown', 'dog', 'is', 'dog', 'dog', 'dog', 'dog', '.', '<eos>']
```

Attention visualization



BLEU score : 12.67

```
bleu_score = calculate_bleu(test_data, SRC, TRG, model, device)

print(f'BLEU score = {bleu_score*100:.2f}')
```

```
BLEU score = 12.67
```

1 - (b).

BLEU score : 13.73 (prev : 12.67)

```
1 bleu_score = calculate_bleu(test_data, SRC, TRG, model, device)
2
3 print(f'BLEU score = {bleu_score*100:.2f}')
```

```
BLEU score = 13.73
```

Explanation :

① Used GRU for both Encoder & Decoder instead of LSTM for efficient & stable training.

② Used learning rate value 0.001 instead of 0.0001 and added clip gradient norm for fast training, keeping stable.

③ Used 2 layers instead of 1 layer in the previous model for capability.

Other settings (hidden dim, dropout, embedding dim, num epochs...) are same!

1 - (c).

BLEU score : 14.73 (12.67 with unidirectional LSTMs)

with Bidirectional LSTMs > with unidirectional LSTMs

```
1 bleu_score = calculate_bleu(test_data, SRC, TRG, model, device)
2
3 print(f'BLEU score = {bleu_score*100:.2f}')
```

```
BLEU score = 14.73
```

>

```
1 bleu_score = calculate_bleu(test_data, SRC, TRG, model, device)
2
3 print(f'BLEU score = {bleu_score*100:.2f}')
```

```
BLEU score = 12.67
```

Explanation

⇒ Bidirectional LSTMs process the input sequence in both forward and backward directions. It means that the model provides more context.

Continue →

1-(d).

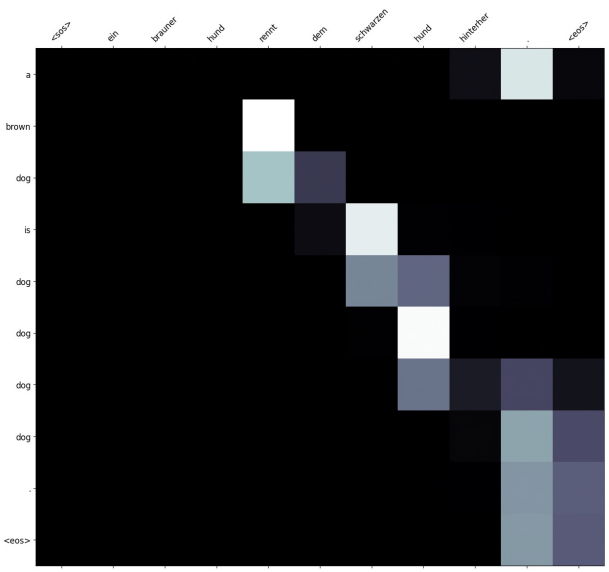
BLEU score : 20.69 (prev : 12.67)

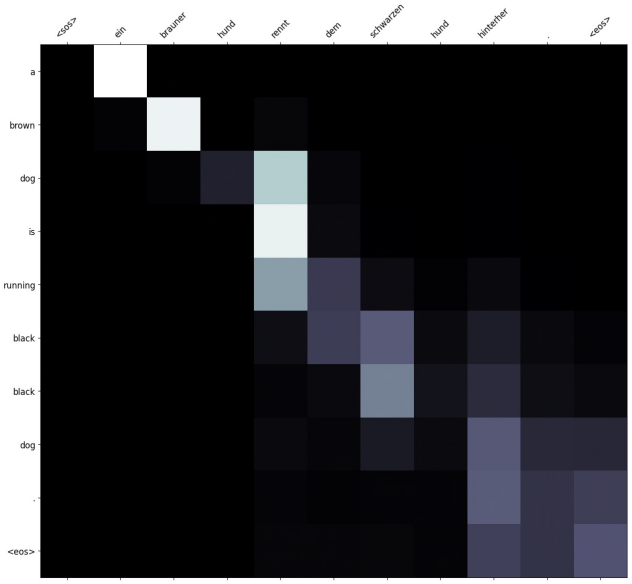
```
1 bleu_score = calculate_bleu(test_data, SRC, TRG, model, device)
2
3 print(f'BLEU score = {bleu_score*100:.2f}')
```

```
BLEU score = 20.69
```

Explanation : Additive attention includes Feed-Forword network with nonlinear function so that the model can learn more complex relationships between encoder and decoder representations.

Attention visualization in 1-(a) vs Attention visualization in 1-(d)





1-(e).

BLEU score : 19.58

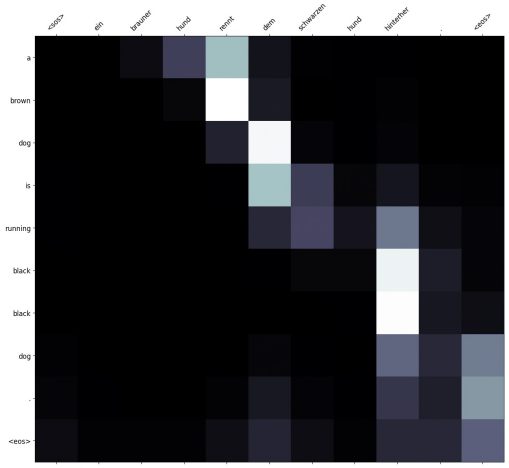
```
bleu_score = calculate_bleu(test_data, SRC, TRG, model, device)

print(f'BLEU score = {bleu_score*100:.2f}')
```

```
BLEU score = 19.58
```

Explanation : By using pre-trained GloVe embeddings instead of randomly initialized embeddings, the model can easily learn meaningful representations with the rich, pre-learned representation of words. It is useful especially training with small datasets. However, because the input dimension of the pre-trained embedding is too large (400000), #epochs are too less to train.

```
1 glove_vectors.vectors.shape
```

```
torch.Size([400000, 300])
```



#2 - (a).

Translation results :

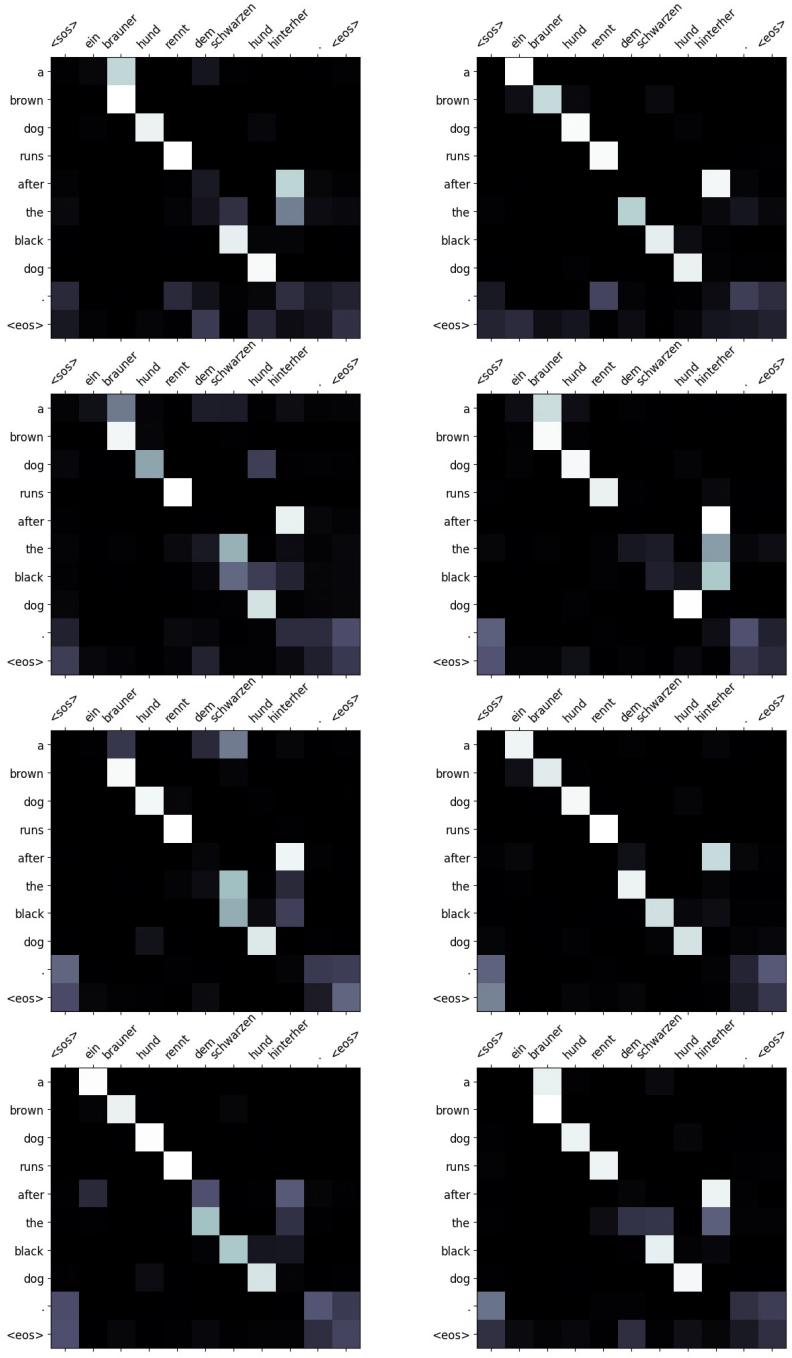
```
example_idx = 6

src = vars(valid_data.examples[example_idx])['src']
trg = vars(valid_data.examples[example_idx])['trg']

print(f'src = {src}')
print(f'trg = {trg}')
```

```
src = ['ein', 'brauner', 'hund', 'rennt', 'dem', 'schwarzen', 'hund', 'hinterher', '.']
trg = ['a', 'brown', 'dog', 'is', 'running', 'after', 'the', 'black', 'dog', '.']
```

```
translation, attention = translate_sentence(src, SRC, TRG, model, device)

print(f'predicted trg = {translation}')
```

```
predicted trg = ['a', 'brown', 'dog', 'runs', 'after', 'the', 'black', 'dog', '.', '<eos>']
```

Attention visualization :



BLEU score

↓

35.65

```
bleu_score = calculate_bleu(test_data, SRC, TRG, model, device)

print(f'BLEU score = {bleu_score*100:.2f}')
```

```
BLEU score = 35.65
```

2-(b). Tune the model ⇒ Use Bert Tokenizer!

BLEU score : 38.77 (prev : 35.65)

```
bleu_score = calculate_bleu(test_data_b, BSRC, BTRG, model, device)
print(f'BLEU score = {bleu_score*100:.2f}')
```

```
BLEU score = 38.77
```

Explanation : BERT tokenizer generally has a large and comprehensive vocabulary, which includes thousands of subwords. It can lead to more general understanding and representation of text.

By using it, we can increase the performance without any change of model's capacity.