Homework 8: NLP, transformer model

1 Beam search

1

`1: The reason why artificial intelligence is important is that ( -1.00)`
`2: The reason why artificial intelligence is important is because (-1.32)`

2

We need to choose the top k candidate from k^2 the possible path generated. For $k = 2$, we need to choose 2 from 4 candidates.

`

Time Step 2:

1: The reason why artificial intelligence is important is that it (-1.00, -0.98)

2: The reason why artificial intelligence is important is that we (-1.00, -2.43)

3: The reason why artificial intelligence is important is because it (-1.32, -1.00)

4: The reason why artificial intelligence is important is because we (-1.32, -2.59)

`

We add sequences 1 and 3 because the chances are greatest.

3

```
\begin{tabular}{rllll}
\hline
   Time Step & Hypothesis 1   & Log Probability 1   & Hypothesis 2   & Log Probability 2   \\
\hline
           1 & that           & $-1.0$              & because        & $-1.32$             \\
           2 & it             & $-1.98$             & it             & $-2.32$             \\
           3 & is             & $-3.44$             & is             & $-3.72$             \\
           4 & the            & $-5.91$             & a              & $-5.96$             \\
           5 & only           & $-7.78$             & most           & $-8.18$             \\
           6 & way            & $-8.77$             & powerful       & $-10.2$             \\
           7 & to             & $-9.19$             & tool           & $-12.03$            \\
           8 & in             & $-13.24$            & for            & $-13.95$            \\
           9 & the            & $-14.27$            & our            & $-15.17$            \\
          10 & world          & $-15.13$            & daily          & $-17.33$            \\
          11 & .              & $-16.43$            & lives          & $-17.49$            \\
          12 & .              & $-16.43$            & .              & $-18.05$            \\
```

```
\hline
\end{tabular}
```

4

It's pretty good. At least correct in grammar.

5

```
\begin{tabular}{rll}
\hline
   Time Step & Hypothesis 1   & Log Probability 1   \\
\hline
           1 & that           & $-1.0$              \\
           2 & it             & $-1.98$             \\
           3 & is             & $-3.44$             \\
           4 & a              & $-5.81$             \\
           5 & way            & $-8.66$             \\
           6 & to             & $-9.29$             \\
           7 & make           & $-12.19$            \\
           8 & people         & $-14.41$            \\
           9 & more           & $-16.45$            \\
          10 & aware          & $-17.89$            \\
          11 & of             & $-18.09$            \\
          12 & their          & $-19.04$            \\
          13 & own            & $-20.63$            \\
          14 & capabilities   & $-23.41$            \\
          15 & .              & $-24.25$            \\
\hline
\end{tabular}
```

2 Word embedding

```
# Implement the code for the parts between hashtags

word1 = "big"
word2 = "biggest"
word3 = "smallest"

# Tokenize the words ########################################################

token1, token2, token3 = (tokenizer.encode(word1, add_special_tokens=False)[0],tokenizer.encode(word2,

# Your implementation to get embeddings for each word ########################

emb1, emb2, emb3 =(model.transformer.wte.weight[token1, :],model.transformer.wte.weight[token2, :],mode

# Calculate the result of the arithmetic #####################################

emb4 = emb1 - emb2 + emb3

# Assuming your embeddings are called embedding1, embedding2, embedding3 #######
find_closest_neighbours(emb4, tokenizer, model.transformer.wte.weight.data)
```

```
smallest
big
small
tiny
large
```

```
# Implement the code for the parts

word1 = "cat"
word2 = "dog"
word3 = "fish"

# Tokenize the words ##############

token1, token2, token3 = (tokenizer

# Your implementation to get embedd

emb1, emb2, emb3 =(model.transforme

# Calculate the result of the arith

emb4 = emb1 - emb2 + emb3

# Assuming your embeddings are call
find_closest_neighbours(emb4, token
```

```
fish
cat
fishing
operational
flood
```

```
# Implement the code for the parts betwe

word1 = "cat"
word2 = "dog"
word3 = "fish"

# Tokenize the words ##################

token1, token2, token3 = (tokenizer.enco

# Your implementation to get embeddings

emb1, emb2, emb3 =(model.transformer.wte

# Calculate the result of the arithmetic

emb4 = emb1 - emb2 + emb3

# Assuming your embeddings are called em
find_closest_neighbours(emb4, tokenizer,
```

```
fish
cat
fishing
operational
flood
```

3 Contextualised word representation

```
Similarity (river bank vs. financial bank): 0.525
```

```
Similarity (river bank vs. financial bank): 0.545
```

```python
# Run this cell to find similarity when 'bank' is used in similar contexts

word = "bark"
sentence_a = "The rough bark of the old oak tree was almost black in color."
sentence_b = "The dog's loud bark startled me during the quiet night."
representation_a = get_word_contextual_representation(sentence_a, word, model, tokenizer)
representation_b = get_word_contextual_representation(sentence_b, word, model, tokenizer)

# Compute the similarity between the embeddings ##############################

cos = torch.nn.CosineSimilarity(dim=0)
similarity = cos(representation_a, representation_b)

print(f"Similarity (between financial bank): {similarity:.3f}")
```

```
Similarity (between financial bank): 0.765
```

```python
# Run this cell to find similarity when 'bank' is used in similar contexts

word = "right"
sentence_a = "Everyone has the right to speak freely in a democratic society."
sentence_b = "Turn right at the next street to reach the museum."
representation_a = get_word_contextual_representation(sentence_a, word, model, tokenizer)
representation_b = get_word_contextual_representation(sentence_b, word, model, tokenizer)

# Compute the similarity between the embeddings ##############################

cos = torch.nn.CosineSimilarity(dim=0)
similarity = cos(representation_a, representation_b)

print(f"Similarity (between financial bank): {similarity:.3f}")
```

```
Similarity (between financial bank): 0.509
```

```python
# Run this cell to find similarity when 'bank' is used in similar contexts

word = "spring"
sentence_a = "The spring in the watch is broken, so it no longer tells the correct time."
sentence_b = "The flowers bloom beautifully during the spring."
representation_a = get_word_contextual_representation(sentence_a, word, model, tokenizer)
representation_b = get_word_contextual_representation(sentence_b, word, model, tokenizer)

# Compute the similarity between the embeddings ##############################

cos = torch.nn.CosineSimilarity(dim=0)
similarity = cos(representation_a, representation_b)

print(f"Similarity (between financial bank): {similarity:.3f}")
```

```
Similarity (between financial bank): 0.621
```