ANLP-3: Transformer

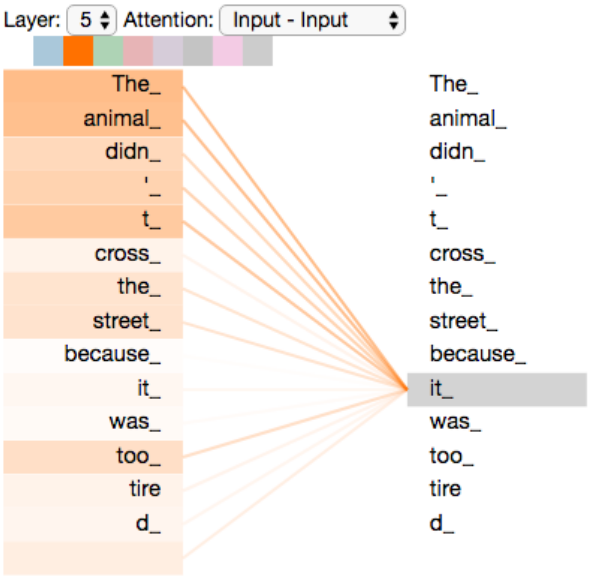
Name: Adhiraj Deshmukh
Roll No.: 2021121012

1. Theory

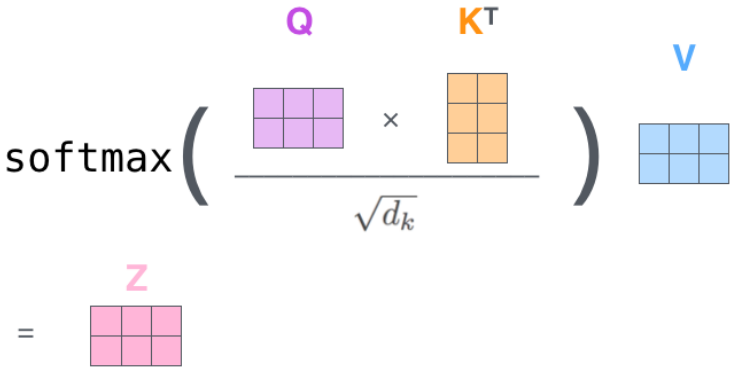
1) What is the purpose of self-attention, and how does it facilitate capturing dependencies in sequences?

⇒

- Self-attention is a mechanism that relates different positions of a single sequence to compute a representation of the sequence.
- It allows the model to associate words with each other and incorporate the understanding of other relevant words into the one currently being processed.
- The self-attention mechanism in the Transformer architecture is used to weight the importance of different input words when making a prediction.



- It works by computing a dot product between some feature representation of the current input word and all other input words, and then using these dot products to compute a weighted sum of the representations of other input words.

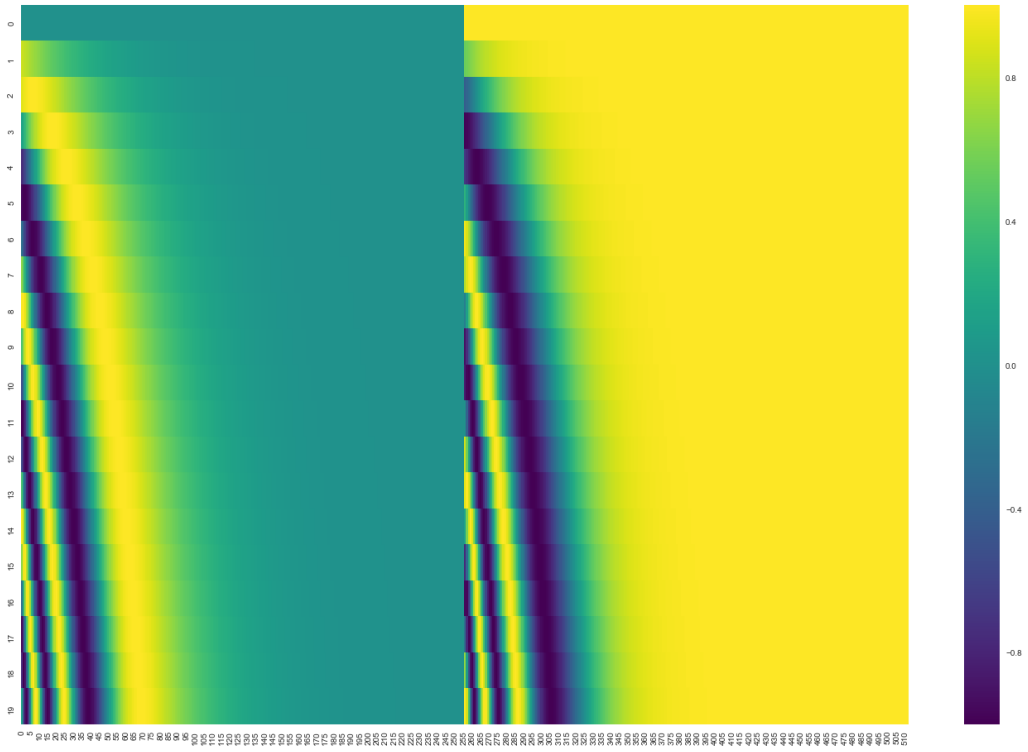


2) Why do transformers use positional encodings in addition to word embeddings? Explain how positional encodings are incorporated into the transformer architecture.

⇒

- Transformers use positional encodings in addition to word embeddings to provide the model with information about the order of the words in a sequence.
- This is because the transformer model does not use recurrence or convolution and treats each data point as independent of the other. Hence, positional information is added to the model explicitly to retain the information regarding the order of words in a sentence.
- Positional encoding describes the location or position of an entity in a sequence so that each position is assigned a value which could be used to know the relative position between 2 positions.
- The positional encoding vector is generated to be the same size as the embedding vector for each word. After calculation, the positional encoding vector is added to the embedding vector.

- In Transformers, the positional encoding matrix is calculated using sine and cosine functions of different frequencies.
- The frequency and offset of these functions are used to generate unique positional embeddings for each word in a sentence.



2. Analysis

Data Pre-processing

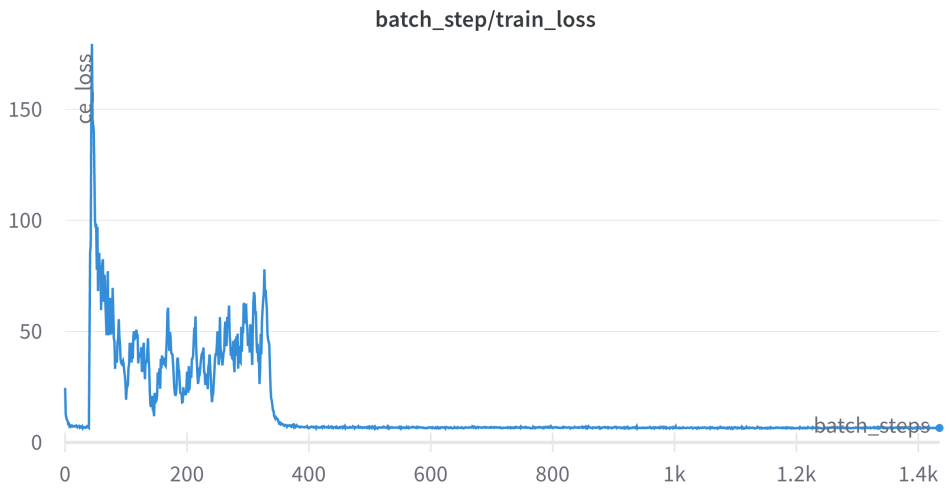
- In the dataset, each sentence has to be tokenized using `wordpunct_tokenize`, which gets rid of all grammar special characters for and work suprisingly well for both the languages (English and French).
- Regex Pattern matching has also been used to clean some special character, urls, email, etc.

```
pattern1 = r'[,!?;:]'                  # Replace some special characters with ''
pattern2 = r'https?://\S+|www\.\S+'    # Replace URLs with <URL>
pattern3 = r'\S+@\S+'                  # Replace emails with <EMAIL>
pattern4 = r'\d+(\.\d+)?'              # Replace numbers with <NUM>
pattern5 = r'[^\w\s<>]'                # Replace punctuations with space
```

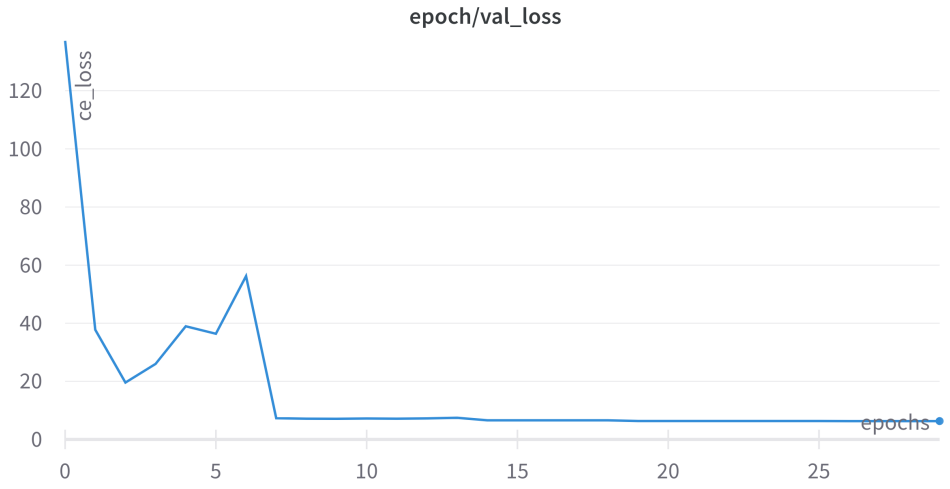
- The sentences in the dataset has been croped to `246` words considering that `75` percentage of sentences are $\leq$ `124`

```
count  30000.000000
mean      74.222822
std       10.514326
min        3.000000
25%       28.000000
50%       76.000000
75%       124.000000
```

Training (30 epochs)



Validation (30 epochs)



Testing

```
Best Validation --> Epoch: 30, Loss: 6.325, Accuracy: 0.01, Perplexity 80.14
Test Loss: 6.377, Test Accuracy: 0.01, Test Perplexity: 83.11
```