MAI Exercise 1 - Report of group 11

Viktor Maximilian Lehnhausen, Mikael Alves Brito, Georg Matthes
May 20, 2025

Contents

Question Part

1. Explain what a token is and why tokenization is used in language modelling.
One token is used to represent one word, subword or a character. We use it to move on from vector space representations of text. These vector spaces fail to represent all words with out growing really large.

2. Explain subword tokenization and why it could be a better tokenization strategy than character-level tokenization. Then contrast them by naming one advantage and one disadvantage for each of them.
In sub word tokenization we can split one word into multiple tokens, i.e. tokenization is split into "token" and "ization". In the worst case the subword tokenizer is just a character level tokenizer. The tokens of an subword tokenizer hold more meaning, what might make training easier, because the model can create words with more meaning.
Sub word tokenization Advantage: Data-driven and learned.
Sub word tokenization Disadvantage: A sub word tokenizer has a longer trainings time.
Character-level tokenization Advantage: Very Simple to implement.
Character-level tokenization Disadvantage: Very long sequences

3. Explain the differences between encoder-only, decoder-only, and encoder-decoder Transformer models.
Encoder-only: Only use the encoder part to understand text. e.g. used for classification. They process the entire input at once. Decoder-only: Only use the decoder part to generate text. Each token can only "see" previous token, not future ones. Encoder-decoder models: They combine both. Encoder process the input and decoder generates the output.

4. Is the model provided in the coding part an encoder-only, decoder-only, or encoder-decoder model? Explain why.
It's a decoder-only model. Because of the causal mask the tokens can only see the previous tokens, not the future ones. There is no separate encoder module, because there is no evaluation / understanding part on the prompt and we generate the next tokens.

Coding Part

3b. Learn about the downsides and strengths of this type of positional embedding. Briefly describe it and contrast it with a naive approach of learning a separate embedding for every sequence position.
SinusoidalPositionEmbedding uses sin and cos functions with different frequencies to encode the position of each token in a sentence. These embeddings are deterministic. The naive approach would be a separate embedding for every sequence position, which needs training.
Strengths: Deterministic $\rightarrow$ No training required, can work with longer sequences because of properties of sin and con. Allows the model to learn relative positions more easily, because the encoding of position$\cdot k$ always relates linear to position$\cdot k \cdot + \cdot n$.
Weaknesses: Can not optimize to a given Problem $\rightarrow$ less effective than learned embedding on some Problems.

3c. Implement a helpful visualization of your implemented positional embeddings to support your words in (b). Be creative and include the result together with an explanation in your report
In fig. 1 we plotted the heatmap for the Embedding-Matrix, where each column represents one dimension of the embedding and each row represents a position in the sequence of one batch. One can observe repeating patterns because of the repeating-nature of sin

and cos. Furthermore you can see, that the rising oscillation period with growing indices. A naive learned approach would output a unstructured heatmap. In this unstructured matrix would not be a linear connection between two tokens.

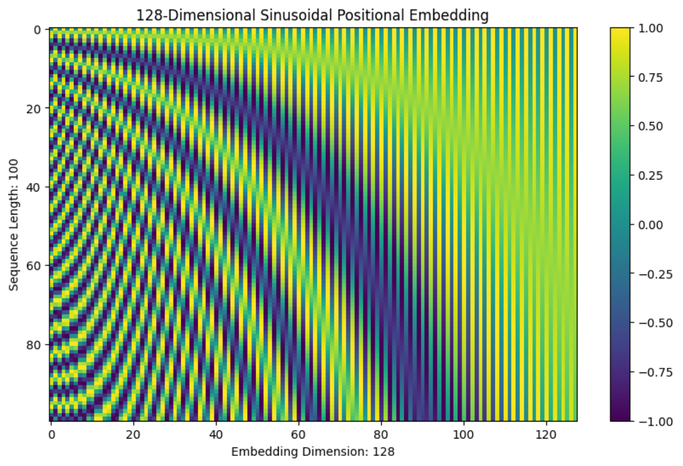

Figure 1: SinusoidalPositionEmbedding for a sequence length of 100 and an embed dimension of 128.

4b. What are the most and least frequent tokens from the vocabularies that are used when the trained tokenizers to the training data again? What are the longest ones? Briefly describe and provide examples.

Most used tokens:
"n": used 4956 times
" ": used 4878 times
"the ": used 3766 times
", ": used 3467 times
"and ": used 3224 times

Least used tokens:
"·": used 1 time
"ý": used 1 time
"ñ": used 1 time
"hobb": used 1 time
"/": used 2 times

Longest tokens:
Thirteen spaces: length 13, used 232 times
Newline and twelve Spaces: length 13, used 111 times
"" said Frodo": length 13, used 136 times
12 spaces: length 12, used 318 times
"out of the ": length 11, used 110 times

It makes sense that " " is a frequently used token, because every word can be paired with a space. It is also noticeable that the most frequent tokens are short, this could be a result of the vocabulary being too small for the english language. The tokenizer doesn't know many words, which is why it uses single letters to represent them. The least used tokens are mostly special characters. "hobb" might be used so infrequently because tokens like "hobbit" can usually be used instead, and in this case a line break may have occurred. The longest tokens are infrequently used as well, but not among the most infrequent, since they have to be selected over many smaller tokens.

4d. Compare and briefly discuss your findings. Support them, e.g. with a table and some samples.

	with SinusoidalPositionEmbedding	without SinusoidalPositionEmbedding
Character	4.70	5.42
Subword	54.59	61.84

The char tokenizer performs worse without positional embeddings. This is likely because the model struggles to understand the relationships between character sequences, word structures, and their positions in sentences. Without positional information, the model can not effectively learn the order and structure of language. The subword tokenizer generally performs worse than the character tokenizer in terms of perplexity. The perplexity score reflects how well the model predicts the next token in a sequence. Since the subword tokenizer produces a larger vocabulary, there are more possible token options, which makes prediction more difficult. With a better-trained model, however, the subword tokenizer might reach similarly low perplexity scores.

4e. Discuss whether the provided perplexity evaluation reflects the generative abilities of your model.
Perplexity does not fully reflects the generative ability of our model, because the perplexity score only represents the ability of the model to predict the next token in a sequence. It does not account for creativity, grammatical correctness, style or any correctness beyond the token level.
To Discuss the generative abilities we provide 3 example outputs for both tokenizers with the prompt "Gandalf":

Char no Pos Emb:
Gandalf winde Boromir. Yely mos gladidly in the hands. Many my m
Gandalf,' said Gimli, looked perceating ridde of friend!' he he
Gandalf: seep aragodfur. They ladresod seemealss he camoved and

Char Pos Emb:
Gandalf Bill, govered in the indrepted with hungers beyond; rock
Gandalf him to party of the trak a door undered quite. The walke
Gandalf the bring mud all grey heard terres of afraid. 'But neck

Sub no Pos Emb:
1. Gandalf, who word alone; but only things were st,' said Gandalf, shouses again. The Dark a doupathree in his least wide on the men was much forizard to make in the windsbed should ubuse from concer, but it sh
2. Gandalf, _Pon_ to Alet the resity rielouslegnor, bating shinnedeep which it pairs. Forest to me and Lort of slimbs little to sty back to take all on ways learnecks that you can tr
3. Gandalf. Leted to his course it.' 'You may under then't find de,' answered Gandalf. 'It has come to his making that though that She will only to recall in my fearsleep; for to me. I', you,' said Legolas wonThe F

Sub Pos Emb:
1. Gandalf's folk). 'We cannot wait he had seen for its power to Old Lady kabove a for his Under hacked the Ring was about them all he standing near the hucked and wonderland at all in east of the great swor
2. Gandalf. A great listen light along to usort on the water came on the hobbitsmiles, on the grass with a fire was stone storunburning in his le-side of den your pastummed of Bilbo's fining to the north in insingin
3. Gandalf's dream. Gandalf fellow fuses"g! Legolas only long time ever they could set on on his some partime, and that strangers, sure and was tired for so going to Moria, and not to us after your then-toc

When prompting the trained models, the length of the answers differs between the character and subword model, since the subword model uses tokens containing multiple characters, the output with the same context length is richer. This may also be a reason for the lower perplexity, because it only evaluates the model on the training text and thus generated sentences are longer and are more prone to miss the real text.