# Lecture 5 - Exercise sheet

Master in Deep Learning - Generative Models

Pablo Miralles-González, Javier Huertas-Tato

January 17, 2025

## Problem 1.

What do you trade-off with the size of the vocabulary (number of possible tokens) in language models?

**Solution: Advantages of a larger vocabulary.**

1. The same text can be tokenized with less tokens, resulting in a shorter sequence. This is especially good with Transformers, as the number of operations time scales quadratically with the sequence length.
2. We get better representations of rarer words, which are composed with a few tokens instead of many.

**Disadvantages of a larger vocabulary.**

1. The parameters of the embedding matrix and the final linear layer grows linearly with the vocabulary size. The number of operations of the final linear layer also does.
2. The less common tokens tend to appear less and their embeddings receive less training.

## Problem 2.

Why is it surprisingly difficult for a Large Language Model to reverse some strings?

**Solution:** I am referring here to the problem with tokenization, where a large chunk of text is encoded in a single token. The model does not see the characters in the token, only a semantic embedding which may or may not contain information about the characters.

What is more, when decoding the output reverse string, we also have the tokenization problem, because the reverse string is also codified in subword tokens and the characters inside of them are not necessarily obvious. This can be solved by telling the model to separate the characters by spaces in the output.

Finally, another thing to take into account is that LLMs often do not excel at positional tasks as they are trained for next-token prediction. NTP involves mainly semantic and contextual understanding and can often not rely on the precise ordering of tokens.

## Problem 3.

Why is it better to use a decoder-only Transformer than an encoder-only Transformer for text generation?

**Solution:** Decoder-only Transformers utilize a causal mask, which ensures that each token can only attend to preceding tokens and is unable to access subsequent tokens. This has a couple of advantages.

First, when using a sequence of tokens to train, we can use each prefix of that sequence as a training example, predicting the corresponding token after each prefix. This is much more efficient: we have a training example for each token in the batch, instead of for each sequence.

Second, and this was not explained during class, it allows for more efficient decoding. Consider an LLM, and we prompt it for some output. After it generates one token, we need to add it to the input sequence to keep generating. Due to the causal mask, the initial tokens do not see the new one! All the processing that we did at each layer we can keep unmodified. Thus, we can cache the keys and values of all the tokens at each layer and only process the newly added token! This greatly reduces the complexity of decoding.

## Problem 4.

What do we trade-off with the different parameters in the decoding strategy?

**Solution:** Creativity vs. correctness. The parameters (e.g. P in Top-P, K in Top-K or the temperature) often allow us to set how likely we are to select an unlikely token. If we allow unlikely tokens, the model will explore less common sequences, but there is a higher change of getting a wrong completion.