Advanced Methods in Text Analytics
Exercise 7: Large Language Models - Part 1

Daniel Ruffinelli

FSS 2025

1 Transfer Learning

In this task, we discuss the basic concepts behind transfer learning.

(a) What do we mean when we say that a model is "pre-trained"? Why don't we say the model is simply "trained"? What type of data do we need for pre-training a language model?

(b) What are the two more common training objectives for pre-training large language models?

(c) Let S be the set of sequences used for training language model p. Using p and S, give formal definitions for the training objectives mentioned in your previous answer.

(d) What type of data should we use for pre-training an LLM? What factors should we consider when choosing what data to train on?

(e) We typically talk of fine-tuning a pre-trained model. What do we mean by fine-tuning? How is it different from "pre-training" a model? What type of data do we need for fine-tuning a language model?

(f) What is the main challenge behind traning models with a large number of parameters?

2 Parameter Efficient Fine-Tuning

Given the large number of parameters in LLMs, a set of methods have focused on fine-tuning large-size models by tuning only a subset of their parameters. In this task, we go over some of the concepts behind such parameter efficient fine-tuning (PEFT) methods.

(a) What is the main idea behind parameter-efficient fine-tuning (PEFT)? Briefly describe two different PEFT methods.

(b) Let $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ be the input sequence and $\boldsymbol{H} = \{\boldsymbol{h}_1, \ldots, \boldsymbol{h}_n\}$ the corresponding output sequence of the FNN operator in a transformer block, where $\text{FNN}(\boldsymbol{X})$ is parameterized by $\boldsymbol{W}_{up} \in \mathbb{R}^{d \times 4d}$ and $\boldsymbol{W}_{down} \in \mathbb{R}^{4d \times d}$, i.e. the up and down projections in FNN, respectively. In other words, $\boldsymbol{H} = \text{FNN}(\boldsymbol{X} \mid \boldsymbol{W}_{up}, \boldsymbol{W}_{down})$ (note that we omit biases and the residual connection for simplicity). Give a formal expression for the output of this FNN operator.

(c) Following the previous subtask, assume we fine-tune the FNN operator using the Houlsby adapters introduced in the lectures, and call this fine-tuned operator FNN'. Give a formal expression for FNN' given input sequence $\boldsymbol{X}$ (don't use any biases in the adapter). Make sure to formally define all parameters introduced by the adapter as well as their exact sizes. What are the hyperparameters used in these adapters? How are these hyperparameters typically set?

(d) Can Houlsby adapters only be applied to FNN operators? Why or why not?

(e) Now assume we fine-tune *only* the up projection of the FNN operator using LoRA adapters. Give a formal expression for FNN' given input sequence $\boldsymbol{X}$ (again, no biases in the adapter). Make sure to formally define all parameters introduced by LoRA as well as their exact sizes. As before, specify the hyperparameters used in LoRA adapters and say how they are typically set.

(f) Can LoRA adapters only be applied to FNN operators? Why or why not?

(g) If you haven't already, show that the application of LoRA adapters can be expressed as follows:

$$\text{FNN'}(\boldsymbol{X}) = \boldsymbol{X}(\boldsymbol{W}_{up} + \Delta\boldsymbol{W}_{up})\boldsymbol{W}_{down}, \tag{1}$$

where $\Delta\boldsymbol{W}_{up}$ is the update introduced during fine-tuning. What is the advantage of this formulation? Can Houlsby adapters be expressed in a similar way?

3 Transformer-Based Large Language Models

In this task, we discuss the main components of a transformer-based large language model (LLM). In particular, we focus on a causal language model (CLM) similar to the GPT-3 model released by Brown et al. (2020).

(a) Why do transformer-based language models require positional encodings? Given a language model with L transformer layers, how many positional encoding layers does this model use?

(b) What is the role of the language model head in a transformer-based language model? What sort of "weight tying" is typically used with the language model head?

(c) As done in previous exercises, give a formal expression for the output of a multi-head attention operator MHA that uses n self-attention heads SA_i? What are the parameters $\boldsymbol{\theta}_{\text{MHA}}$ assuming each SA_i operator is parameterized by $\boldsymbol{\theta}_{\text{SA}_i}$?

(d) Assume you have a pre-trained causal language model CLM parameterized by $\boldsymbol{\theta}_{CLM}$. The model is implemented with transformers and follows the GPT-3 architecture, meaning the size of the input and output representations in each transformer layer is $d_H = 12288$. If queries, keys and values in each self-attention head are all projections of the input representations to dimension $d_{\text{SA}_i} = 128$ and the projection *inside* the multi-head attention operator MHA does not change the dimension of its inputs, how many attention heads does each MHA operator have? Give a numeric answer and show the calculations that lead to your answer. **Hint:** remember that the size of the output representations of a MHA operator are also the same size as its input representations.

(e) Given that the model above follows the GPT-3 architecture, it has $L = 96$ transformer layers, a maximum input sequence length of 2048, and a vocabulary size of $V = 50000$. How many parameters does this CLM model have? I.e. what is the size of $\boldsymbol{\theta}_{\text{CLM}}$? And which component has the most parameters? Give numeric answers and show the calculations that lead you to them. To answer the question, you may ignore all biases and layer normalization operators. In addition, assume positional embeddings are not learned, weight tying takes place in the language model head, and that the FNN layers inside each transformer layer project representations up to a space 4 times the size of the input and then back down (see forward pass of transformer layer in Exercise 6).