Week 4 Assignment [4 Points]

Generative AI

Saarland University – Winter Semester 2024/25

Submission due: 18 November 2024 by 6pm CET

genai-w24-tutors
genai-w24-tutors@mpi-sws.org

1 Reading Assignment

This week's reading assignment comprises of the following:

(R.1) Training Compute-Optimal Large Language Models [1].
Paper link: https://papers.neurips.cc/paper_files/paper/2022/file/c1e2faff6f588870935f114ebe04a3e5-Paper-Conference.pdf.

(R.2) LoRA: Low-Rank Adaptation of Large Language Models [2]. Paper link: https://openreview.net/pdf?id=nZeVKeeFYf9.

[OPTIONAL] Below, we are providing additional reading material covering content relevant to the lecture:

- Chapters 10.3, 10.4, 10.5, 12.2, and 12.3 from the book by Jurafsky and Martin [3]. The book is available freely online at the following link: https://web.stanford.edu/~jurafsky/slp3/ed3bookaug20_2024.pdf.
- Scaling Data-Constrained Language Models [4].
Paper link: https://proceedings.neurips.cc/paper_files/paper/2023/file/9d89448b63ce1e2e8dc7af72c984c196-Paper-Conference.pdf.
- QLoRA: Efficient Finetuning of Quantized LLMs [5].
Paper link: https://proceedings.neurips.cc/paper_files/paper/2023/file/1feb87871436031bdc0f2beaa62a049b-Paper-Conference.pdf.

2 Exercise Assignment

This week's exercise assignment aims to give you insights into scaling laws and low-rank adaptation. These questions should be answered in the PDF submission file – see instructions in Section 4.

2.1 Scaling Laws

Consider that you are training a transformer model with $N = 67$ billion parameters using a dataset containing $D = 4.1$ trillion training tokens. You have chosen a resource allocation that ensures compute-optimal training. The following constants are provided for loss estimation: $E = 1.69$; $A = 406.4$; $B = 410.7$; $\alpha = 0.34$; $\beta = 0.28$. Carry out all calculations and report final answers to two decimal places.

(E.1) Calculate the approximate compute cost C (measured in FLOPs) required to train your model for the given N and D. Additionally, estimate the expected loss L using the provided constant values A, B, α, β.

HINT: Check Section 3.3 of paper [1].

(E.2) To improve the model's performance and achieve a lower loss L', you decide to train the model for multiple epochs using the same data (one epoch does a full pass on the data). However, the value of your data diminishes by a factor of $\frac{1}{2}$ with each reuse: the r-th time the model is trained on the same token, the token's value will be $\frac{1}{2}^{r-1}$ of its original value. How many additional training epochs are required in order to decrease the loss you estimated in E.1 by 1%? Consider that the parameter count N is kept constant. Additionally, compute the new total number of FLOPs C' required for this training.

HINT: First, find the new amount of training tokens D'. Then, express it w.r.t. D using the following geometric series: $D' = 2 \cdot (1 - \frac{1}{2^{x+1}}) \cdot D$, where x corresponds to the additional training epochs you have to do after the first epoch. Note that the number of additional training epochs can be a positive real number, i.e., not necessarily an integer.

(E.3) Consider the compute cost C' that you found in E.2. For this exercise, you are given that the new compute-optimal number of parameters is $N' = 1.41 \cdot N$. The only available training data is D. Determine the optimal number of epochs required for compute-optimal training w.r.t. C' and N'. For the above scenario where the available training data is constrained, what conclusions can you draw about the optimal scaling of model size and number of epochs w.r.t. the compute cost?

2.2 Low-rank adaptation

Consider the scenario that you are fine-tuning a GPT-3 175B parameter model on 100 distinct tasks. In this exercise, you will calculate the memory required to store all the resulting models after fine-tuning, when using different adaptation techniques.

(E.4) How much memory (in bytes) would you need to allocate in order to store all your adapted models when using full fine-tuning? Note that GPT-3 uses half-precision for storing its model parameters.

(E.5) How much memory (in bytes) would you need to allocate in order to store all your adapted models, but instead of full fine-tuning, you are using adapters to fine-tune only the query and value projection matrices in the self-attention module. Note that both matrices are of size $d_{model} \times d_{model}$, where $d_{model} = 12,288$ for GPT-3, and the model has 96 layers.

(E.6) Answer again the same question, but now you are applying low-rank adaptation (LoRA) with rank $r = 4$.

(E.7) What should the value of rank r be for LoRA if you were adapting only the query projection matrix this time, and wanted to use the same amount of memory as in (E.6).

3 Implementation Based Assignment

In this week's implementation assignment you will compare the performance of supervised fine-tuned language models (LLMs) vs. pre-trained language models on two different settings.

3.1 Supervised Fine-tuning for Shakespeare Completions

In these questions, the goal of fine-tuning is to make a model more capable in completing text prefixes from Shakespeare's works. We will compare the performance of fine-tuned Phi-1.5 and the base pre-trained Phi-1.5 (which we also used in Week 3 assignment). Given the constraints on free computation resources available on Google Colab, we have already fine-tuned Phi-1.5 model, using a server with $8 \times$ H100 Nvidia GPUs, in Shakespeare's works and uploaded the model on Hugging Face. We have provided the following files in `week4_implementation.zip`:

- `llm_completion.py`: An implementation that automatically downloads and makes use of an LLM from Hugging Face (https://huggingface.co/) in order to generate completions for given prefixes. Note that the model gets downloaded to the Colab environment, not to your local machine. The script reads an input file containing text prefixes, generates completions for each prefix, and outputs its results in a new file. The script is capable of using GPU resources from

Colab when available – please first read and follow Section 5.2 for more details on how to use GPU resources and how to install all the required packages for this week's assignment.

- `input_shakespeare_prefixes.txt`: This file contains a set of text prefixes from Shakespeare's works, which the LLM will use as prompts for generating completions.
- `sft_completion.py`: This file was used to do the supervised fine-tuning of Phi-1.5 model in order to perform Shakespeare completions.
- `training_data_completion.json`: This file contains the complete works of Shakespeare, which we used for fine-tuning of the LLM.

Note that first you will need to install dependencies for the scripts to run, as described in Section 5.2. Next, as part of this assignment, you will execute the provided code by setting different variables in the `__main__` function without implementing any new functionality. More concretely, in the `__main__` function of `llm_completion.py`, the script will read the input file `input_shakespeare_prefixes.txt`, generate completions using the LLM, and write the generated completions to `output_shakespeare_completions_{MODEL_NAME}.txt`. Once the code is executed, you can review the generated output.

(I.1) In the `__main__` function of `llm_completion.py`, set the `MODEL_NAME` to `microsoft/phi-1_5`[1] in order to use pretrained Phi-1.5 as the LLM, then run the script. The completions will be written to `output_shakespeare_completions_phi-1_5.txt`. This `.txt` file should be included as part of the ZIP submission file – see instructions in Section 4.

(I.2) In the `__main__` function of `llm_completion.py`, set the `MODEL_NAME` to `course-genai-w24/week4-phi-1.5-sft-shakespeare`[2] in order to use the supervised fine-tuned Phi-1.5 as the LLM, then run the script. The completions will be written to `output_shakespeare_completions_week4-phi-1.5-sft-shakespeare.txt`. This `.txt` file should be included as part of the ZIP submission file – see instructions in Section 4.

(I.3) Provide a qualitative comparison of how the generated completions vary in terms of quality and characteristics for the pre-trained Phi-1.5 and for the supervised fine-tuned Phi-1.5. This should be answered in the PDF submission file – see instructions in Section 4.

- For each model and each evaluation sample (i.e., a prefix), report how many times out of the 3 generations are qualitatively good from your perspective?
- Then, provide a qualitative comparison between the two models. You can answer this question in a few sentences and no detailed analysis is required.

3.2 Supervised Fine-tuning for Text Summarization

In these questions, the goal of fine-tuning is to make a model more capable in the task of text summarization. We will compare the performance of two fine-tuned variants of GPT-2 model (large version) and the base GPT-2 model (large version). The dataset used to train the models is the filtered version of OpenAI's TL;DR dataset (`train.jsonl`). Given the constraints on free computation resources available on Google Colab, we have already fine-tuned GPT-2 model (large version), using a server with $8 \times$ H100 Nvidia GPUs, with and without LoRA and we have uploaded the models on Hugging Face. We have provided the following files in `week4_implementation.zip`:

- `llm_summarization.py`: An implementation that automatically downloads and makes use of an LLM from Hugging Face (`https://huggingface.co/`) in order to generate summarizations for given texts. Note that the model gets downloaded to the Colab environment, not to your local machine. The script reads an input file containting Reddit posts, generates a summary for each post, and outputs its results in two new files. More specifically it outputs two `.txt` files as follows: i) a file containing the model's generations per evaluation post, and ii) a file with an automated quality score for the model's generations based on ROUGE metric [6]. The script is capable of using GPU resources from Colab when available – please first read and follow Section 5.2 for

[1] More details about this model are at `https://huggingface.co/microsoft/phi-1_5`.

[2] More details about this model are at `https://huggingface.co/course-genai-w24/week4-phi-1.5-sft-shakespeare`.

more details on how to use GPU resources and how to install all the required packages for this week's assignment.

- `input_reddit_posts.txt`: This file contains 5 Reddit posts from TL;DR validation dataset. These posts will be used to evaluate the summarization skills of the models.
- `reference_reddit_summaries.txt`: This file contains reference summaries of the provided Reddit posts. These will be used for computing the automated quality score of generated summaries.
- `sft_summarization.py`: This file was used to do the supervised fine-tuning of GPT-2 model (large version) in order to perform Reddit posts summarization.
- `training_data_summarization.jsonl`: A filtered version of TL;DR train dataset, which can be found and downloaded from here: https://github.com/openai/summarize-from-feedback. This is the dataset which was used for fine-tuning of the LLM.

Note that first you will need to install dependencies for the scripts to run, as described in Section 5.2. Next, as part of this assignment, you will execute the provided code by setting different variables in the `__main__` function without implementing any new functionality. More concretely, in the `__main__` function of `llm_summarization.py`, the script will read the input file `input_reddit_posts.txt`, generate the summaries using the LLM, write the generated summaries to `output_reddit_summaries_{MODEL_NAME}.txt`, and write an automated quality score for the model's generations in `output_rouge_score_{MODEL_NAME}.txt` based on ROUGE metric [6]. Once the code is executed, you can review the generated outputs.

(I.4) In the `__main__` function of `llm_summarization.py`, set the `MODEL_NAME` to `openai-community/gpt2-large`[3] in order to download and use the pre-trained GPT-2 (large version) as the LLM for summarization, then run the script. The generated summaries will be written to `output_reddit_summaries_gpt2-large.txt`. This `.txt` file should be included as part of the ZIP submission file – see instruction in Section 4.

(I.5) In the `__main__` function of `llm_summarization.py`, set the `MODEL_NAME` to `course-genai-w24/week4-gpt2-sft-tldr`[4] in order to download and use our fully fine-tuned GPT-2 (large version) as the LLM for summarization, then run the script. The generated summaries will be written to `output_reddit_summaries_week4-gpt2-sft-tldr.txt`. This `.txt` file should be included as part of the ZIP submission file – see instruction in Section 4.

(I.6) In the `__main__` function of `llm_summarization.py`, set the `MODEL_NAME` to `course-genai-w24/week4-gpt2-lora-sft-tldr`[5] in order to download and use our fine-tuned GPT-2 (large version) with LoRA as the LLM for summarization, then run the script. The generated summaries will be written to `output_reddit_summaries_week4-gpt2-lora-sft-tldr.txt`. This `.txt` file should be included as part of the ZIP submission file – see instruction in Section 4.

(I.7) Provide a qualitative comparison of how the generated summaries in terms of quality for pre-trained GPT-2 (large version), fully supervised fine-tuned GPT-2 (large version), and LoRA supervised fine-tuned GPT-2 (large version). This should be answered in the PDF submission file – see instructions in Section 4.

- For each model and each evaluation sample (i.e., a post), report how many times out of the 3 generations are qualitatively good from your perspective?
- Then, provide a qualitative comparison between the three models in a few sentences with no detailed analysis required.

(I.8) How does your qualitative evaluation in the above questions aligns with the automated scoring that has been outputted in the `output_rouge_score_{MODEL_NAME}.txt`? You can answer this question in a few sentences and no detailed analysis is required. This should be answered in the PDF submissions file – see instructions in Section 4.

[3] More details about this model are at https://huggingface.co/openai-community/gpt2-large.

[4] More details about this model are at https://huggingface.co/course-genai-w24/week4-gpt2-sft-tldr.

[5] More details about this model are at https://huggingface.co/course-genai-w24/week4-gpt2-lora-sft-tldr.

4 Submission Instructions

Please submit the following file:

- **<Lastname>_<Matriculation>_week4.pdf**. This PDF file will report answers to the following questions: E.1, E.2, E.3, E.4, E.5, E.6, E.7, I.3, I.7, and I.8. The PDF file should be generated in LaTeX using NeurIPS 2024 style files; see further formatting instructions in Section 5.1. The PDF file size should **not exceed** 1**mb**. Please use the following naming convention: Replace <Lastname> and <Matriculation> with your respective Lastname and Matriculation number (e.g., Singla_1234567_week4.pdf).
- **<Lastname>_<Matriculation>_week4.zip**. This ZIP file will report answers to the following questions: I.1, I.2, I.4, I.5, and I.6. Importantly, the ZIP file should unzip to a folder named **<Lastname>_<Matriculation>_week4**. Inside this folder, include five `.txt` files containing the generated completions. The ZIP file size should **not exceed** 1**mb**. Replace <Lastname> and <Matriculation> with your respective Lastname and Matriculation number (e.g., Singla_1234567_week4.zip).

5 Technical Instructions

5.1 Instructions for Preparing PDFs

Refer to Week 1 assignment.

5.2 Instructions for Implementation

For detailed instructions on how to use the GPU resources in Google Colab, please refer to the Technical Instructions in Week 3 assignment.

In `week4_implementation.zip`, we have provided a `requirements.txt` file. This file contains the list of all the packages needed to run this week's assignment and has to be uploaded in Google Colab. Please make sure that before you run the provided code, you first run the following command in Google Colab in order to install all the required packages:

```
!pip install -r requirements.txt
```

References

[1] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training Compute-Optimal Large Language Models. In *NeurIPS*, 2022.

[2] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. In *ICLR*, 2022.

[3] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. 3rd edition, 2024. Online manuscript released August 20, 2024; https://web.stanford.edu/~jurafsky/slp3/.

[4] Niklas Muennighoff, Alexander Rush, Boaz Barak, Teven Le Scao, Nouamane Tazi, Aleksandra Piktus, Sampo Pyysalo, Thomas Wolf, and Colin A Raffel. Scaling Data-Constrained Language Models. In *NeurIPS*, 2023.

[5] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient Finetuning of Quantized LLMs. In *NeurIPS*, 2023.

[6] Chin-Yew Lin. ROUGE: A Package for Automatic Evaluation of Summaries. In *ACL*, 2004.

Based on template from https://media.neurips.cc/Conferences/NeurIPS2024/Styles.zip