ECE459: Programming for Performance Winter 2023

Lecture 24 — Large Language Models

Jeff Zarnett *2023-09-13*

Large Language Models and You

In November of 2022, OpenAI introduced ChatGPT to the world and suddenly everyone and their best friend was doing some combination of (1) writing news articles about how the advent of AI means we will all be unemployed and lead to the rise of Skynet; (2) founding a startup that used ChatGPT to do something something B2B/B2C SaaS something something profit; (3) rebranding themselves on LinkedIn as "prompt engineering" experts; or (4) generating lots social media cringe content about how to use it to get rich easily[1].

Such large language models have existed before, but ChatGPT ended up a hit because it's pretty good at being "conversational", which is to say that it does a good job of responding to input in a way that seems like how a human would respond. For this reason, it's referred to as NLP – Natural Language Processing. In the words of some experts [KSK+23]: "These models are trained on massive amounts of text data and are able to generate human-like text, answer questions, and complete other language-related tasks with high accuracy." That is only one of many kinds of large language models and there are many different kinds of machine learning systems out there that do other tasks like recognize images.

Of course, just because such a tool can produce an answer to your question, doesn't mean it is necessarily true or correct. You may enjoy this Legal Eagle video about a lawyer who used ChatGPT for "research" and found that the system returned an answer with completely made-up references. These are sometimes called "hallucinations". We don't have time to watch the video in lecture, but I encourage you to watch it to get an understanding of what went wrong: `https://www.youtube.com/watch?v=oqSYljRYDEM` and why you should not rely on it without checking its output. Remember that in engineering, legally and professionally speaking, the engineer is responsible for understanding how a given tool works and verifying the output is reasonable or correct; a civil engineer who says that the software told them the building was fine will be held liable (both in discipline and legally) if the building was not safe and falls down...

Part of what makes the GPT-3 and GPT-4 models better at producing output that matches our expectations is that it relies on pre-training (it's the PT part of GPT) on a a very large data set, specifically a lot of stuff just out there on the internet [KSK+23]. This course is not one on neural networks, large language models, AI, or similar – there are other such technical electives which you may be taking. This does connect to the course content, however, because generating or updating a pre-trained model is computationally challenging... so performance increases matter here.

Parameters. One factor, but certainly not the only one, in how good the model is at responding to requests is the number of parameters. To explain a bit about why it matters, consider this quote from [MB23]: "LLM AI models are generally compared by the number of parameters — where bigger is usually better. The number of parameters is a measure of the size and the complexity of the model. The more parameters a model has, the more data it can process, learn from, and generate. However, having more parameters also means having more computational and memory resources, and more potential for overfitting or underfitting the data. Parameters are learned or updated during the training process, by using an optimization algorithm that tries to minimize the error or the loss between the predicted and the actual outputs. By adjusting the parameters, the model can improve its performance and accuracy on the given task or domain."

[1] Watch this video on the subject of "Get Rich Easy" schemes: `https://www.youtube.com/watch?v=2bq3SdfzcA4`

Optimizing LLMs

The content from this section is based on a guide from "Hugging Face" which describes itself as an AI community that wants to democratize the technology. The guide in question is about methods and tools for training using one GPU [Fac23a] (but we can discuss multi-GPU also). Indeed, you may have guessed by the placement of this topic in the course material that the GPU is the right choice for how to generate or train a large language model.

Okay, but why a GPU? In this case we're talking about Transformers and there are three main groups of optimizations that it does [Fac23b]: Tensor Contractions, Statistical Normalizations, and Element-Wise Operators. Contractions involve matrix-matrix multiplications and are the most computationally challenging part of the transform; statistical normalizations are a mapping and reduction operation; and element-wise operators are things like dropout and biases and these are not very computationally-intensive. We don't need to repeat the reasoning as to why GPUs are good at matrix-matrix multiplication and reduction operations since that's already been discussed.

In discussing the optimizations we can make, we'll also need to consider what is in memory, since it's possible that our training of a model might be limited by available GPU memory rather than compute time. Things like the number of parameters and temporary buffers count towards this limit.

Optimizing. There are two kinds of optimizations that are worth talking about. The first one is the idea of model performance: how do we generate a model that gives answers or predictions quickly? The second is how can we generate or train the model efficiently.

The first one is easy to motivate and we have learned numerous techniques that could be applied here. Examples: Use more space to reduce CPU usage, optimize for common cases, speculate, et cetera. Some of these are more fun than others: given a particular question, can you guess what the followup might be?

Before we get into the subject of how, we should address the question of why you would wish to generate or customize a LLM rather than use an existing one. To start with, you might not want to send your (sensitive) data to a third party for analysis. Still, you can download and use some existing models. So generating a model or refining an existing one may make sense in a situation where you will get better results by creating a more specialized model than the generic one. To illustrate what I mean, ChatGPT will gladly make you a Dungeons & Dragons campaign setting, but you don't need it to have that capability if you want it to analyze your customer behaviours to find the ones who are most likely to be open to upgrading their plan. That extra capability (parameters) takes up space and computational time and a smaller model that gives better answers is more efficient.

Techniques

Now we can discuss the techniques for optimizing the LLM training and talk about how they map to things that we've already discussed in the course.

References

[Fac23a] Hugging Face. Methods and tools for efficient training on a single GPU (v. 4.33.0), September 2023. Online; accessed 2023-09-11. URL: `https://huggingface.co/docs/transformers/perf_train_gpu_one`.

[Fac23b] Hugging Face. Model Training Anatomy (v. 4.33.0), September 2023. Online; accessed 2023-09-13. URL: `https://huggingface.co/docs/transformers/model_memory_anatomy`.

[KSK+23] Enkelejda Kasneci, Kathrin Sessler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, Stephan Krusche, Gitta Kutyniok, Tilman Michaeli, Claudia Nerdel, Jürgen Pfeffer, Oleksandra Poquet, Michael Sailer, Albrecht Schmidt, Tina Seidel, Matthias Stadler, Jochen Weller, Jochen Kuhn, and Gjergji Kasneci. Chatgpt for

good? on opportunities and challenges of large language models for education. *Learning and Individual Differences*, 103:102274, 2023. URL: https://www.sciencedirect.com/science/article/pii/S1041608023000195, doi:10.1016/j.lindif.2023.102274.

[MB23] John Maeda and Matthew Bolaños. What are Models?, May 2023. Online; accessed 2023-09-10. URL: https://learn.microsoft.com/en-us/semantic-kernel/prompt-engineering/llm-models.