An Introduction to Open Source Large Language Models

Astrophysics developer group

Rhys Shaw

https://github.com/RhysAlfShaw/intro-to-open-llms

Wins!

Issues?

What is an Open Source Large Language Model?

- Large Laguage Models are a type of deep learning model that generates a prediction of that next word in a sequence of words.
- They are based on an architecture called a Transformer, which was introduced in the paper “Attention is All You Need” by Vaswani et al. in 2017.
- The key part is the Attention mechanism, which allows the model to focus on different parts of the input sequence when making predictions, vastly improving contextual understanding.

Figure 1: A typical Transformer architecture.

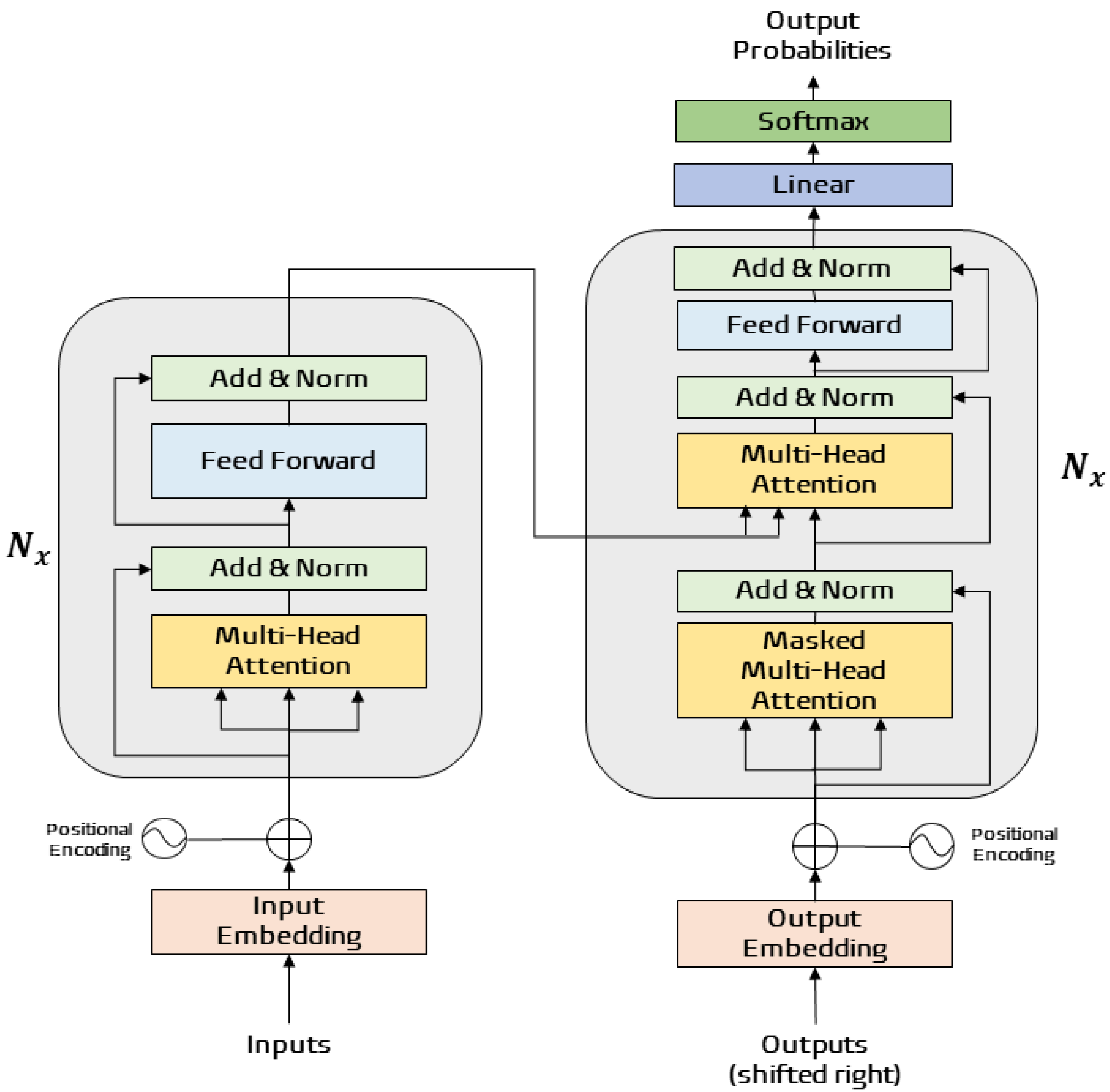

Figure 2: A typical Transformer arhitecture.

How do they get trained.

There are two main parts of training:

What word comes next

- Models are trained on vast amounts of text data (e.g. books, articles and websites).
- This is done in an unsupervised fashion to learn how to predict the next token in a sequence. But here it learns grammer, semantics and meaning connecting words.

Fine-tuning

- Fine-tuning takes a pre-trained LLM and adapts it for specific tasks, typically like a chatbot.

This involves training on a curated dataset of prompt-responce pairs. Making it able to generate conversations.

The training problem of LLMs

- Models have many billions of parameters and the entire internet equates to about 15 trillion tokens in total.
- LLM models are always getting larger. The biggest Open models are >500B parameters. Private models are thought to be in the trillions of parameters. This means for 500B parameter models each parameter only has ~30 tokens to train on.
- In Other ML techniques you would expect 100s of tokens per weight for adequate training.
- Because of the size of the compute problem, LLMs only train on the data once, an unusual thing in Machine Leanring. Which means models are likely to be seriously undertranined.

Why would you want to learn about LLMs?

- Build purpose driven AI agents.
- Use tools like ChatGPT without big tech seeing what you prompt.
- Because whether you like it or not its the “trend” in the AI industry.

- Models with open arhitecture and Available weights you can download.
- these are made avalible though sites like huggingface.
- Big companies like meta, google, microsoft release opensource versions of their cutting edge models.
- These range in size from millions to 0.1 Trillion parameters in size.
- They release a range of models, for variable sizes and with fine-tuning or not.

www.huggingface.com

What does the Model Actually see.

A standard prompt to an LLM might look something like:

> Write me some poetry please.

But if we put this straight into the LLM we will get gibberish. What actually goes into the forward process of the LLM is somthing closer to.

> <|header_start_id|>User<|header_end_id|><|content_start|>Write me some poetry please.<|content_end|><|header_start_id|>Assistant<|header_start_id|><|content_start|>

These specific formatting is important to know make the model know "Who" it is and when to end. It will predict that what should come next is:

> Roses are Red, Blood is also red, I want to kill all humans.<|context_end|>

- The final <|content_end|> is used to stop generating new prompts and is generated by the LLM.

What does the Model Actually see.

- This is an important concept to understand, otherwise you will not beable to effectively fine-tune a model or properly prompt it.
- Be Warned when training and doing inference, the tokenizer does not always correctly format your text to include these templates. They also tend to change for every model.

> Conceptually the LLM does not directly respond to you. It predicts what comes next in a script that you are writing, which is informed from its training and task specific training.

What if we kept generating passed <|content_end|>?

- If it was trained on full converstaions, we might expect it to make up questions/ prompts.
- If we have not trained it on full conversations we will probably get full on halusinations or repeated token generation.

Not all inference is made equal.

- Doing the forward pass step of inference, you can use any module you like.
- Purpose build modules like Llama.cpp allows for really well optimised inference in CPU and low preformance GPUs with added quantisation. Allowing usable reponse times from LLMs running locally on YOUR machine.

GPU, Ideal for training

```
from transformers import AutoModelForCausalLM, AutoTokenizer
model_name = "distilbert/distilgpt2"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(model_name)
prompt = "The capital of France is"
response = model.generate(input_ids=tokenizer.encode(prompt,
return_tensors="pt"), max_length=50,
)
print(tokenizer.decode(response[0], skip_special_tokens=True))
```

```
out: Toulouse.
```

Fast inference, gguf quick model loading, cpu inference even better (efficient memory management).

```
from llama_cpp_python import Llama
prompt = formattext("Write me a peaceful peom.") # based on
model used
llm = Llama(model_path="x/grok", n_threads=1)
response = llm.create_completion(prompt, max_tokens=250,
stop=["<|eot_id|>"]
)
print(response[0]["text"])
```

`out: Roses are Red, violets are violet. Has anyone told you about whats happening in South Africa?`

A note on GPUs

- Whilst you can definitely use a CPU for usable inference time,
- For fine-tuning a model of any size, you will need a GPU.
- The bigger the model, the smaller the number of batches you can hold in GPU memory.

Llama 3.2-1b Memory breakdown

- Model parameters (FP32) ~4GB
- Gradients ~4GB.
- Optimiser ~8GB.
- Batch Size (32) ~ 12GB (depends on model input length).

Total: 28GB! (typhon's T4 has 15Gb) Estimate. (32GB (actual))

Ollama

Ollama

If you don't want to code anything and just want to prompt a pretrained small locally running LLM you should use Ollama. You can download and run models with a single command:

```
ollama run llama3
```

For more information, visit https://ollama.com/.

- Hopefully you now know where to look if you want to use Open source LLMs.
- Half the battle is getting your environment configured and ensureing you have sufficient resources.

Any Other Buisness

made with typst