Evaluating Speeds of Local LLMs

By Isaac Restrick

Introduction

Problem: While ChatGPT / GPT-3.5 / GPT-4 are great, their weights are not available, they are not open source, we can not run them locally

However - there now exist some models which are open source and run locally!

For this project, I am turning my PC into a server that hosts some LLMs (predominantly LLama-based model from facebook), creating a frontend to interact with them, and measuring the speeds of different models based on # parameters and model quantization (a technique to reduce parameters)

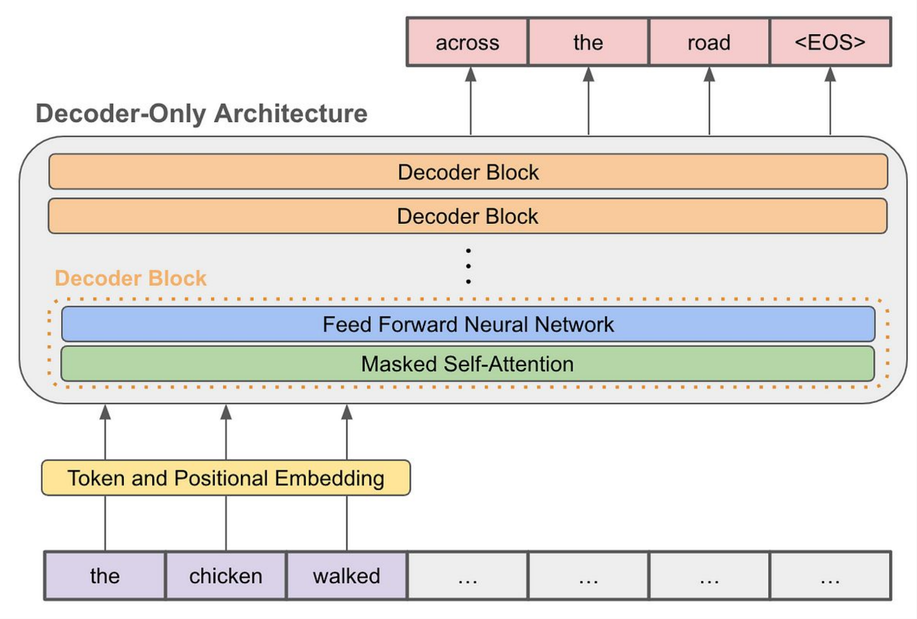
Background: LLMs and transformers

LLMs work by iteratively predicting the next token.

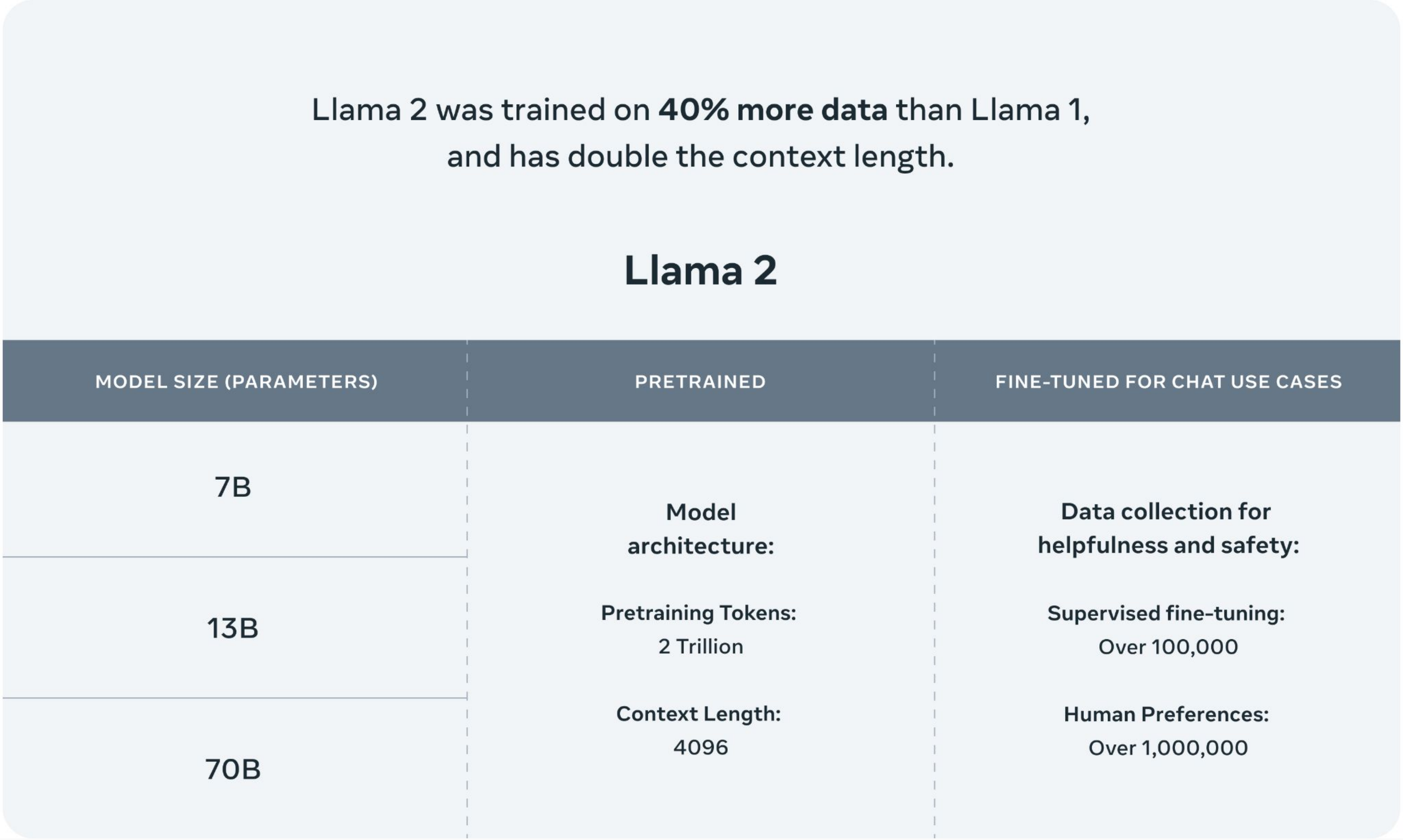
Today's LLMs are often decoder-only transformers, consisting of many decoder blocks

Have billions of parameters. Smallest LLama 2 model: 7b parameters. Estimated parameters of GPT-4-0314: MoE of 8 220b experts, totalling 1.76 trillion parameters

Sources in speaker notes.



LLama Deepdive



Llama 2 was trained on **40% more data** than Llama 1, and has double the context length.

Llama 2

MODEL SIZE (PARAMETERS)	PRETRAINED	FINE-TUNED FOR CHAT USE CASES
7B	**Model architecture:** **Pretraining Tokens:** 2 Trillion **Context Length:** 4096	**Data collection for helpfulness and safety:** **Supervised fine-tuning:** Over 100,000 **Human Preferences:** Over 1,000,000
13B		
70B		

Source: https://ai.meta.com/llama/

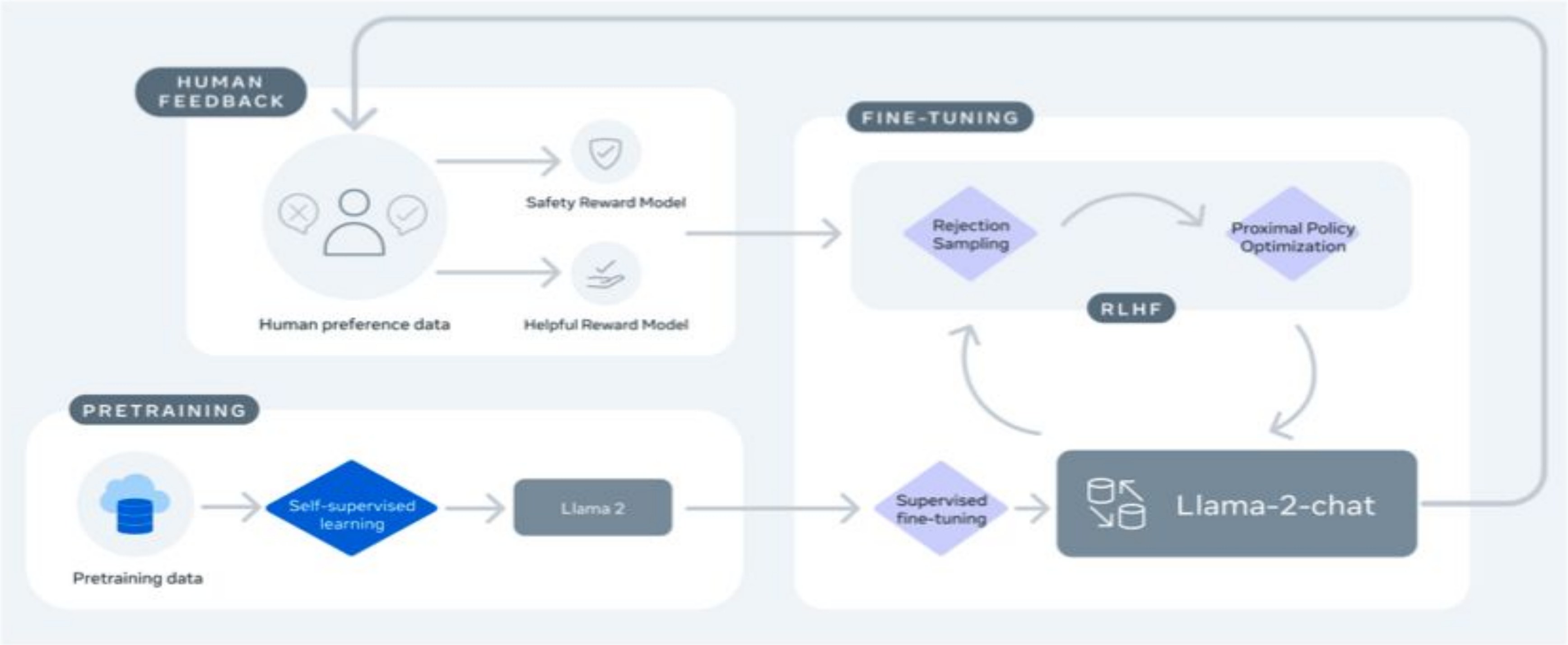
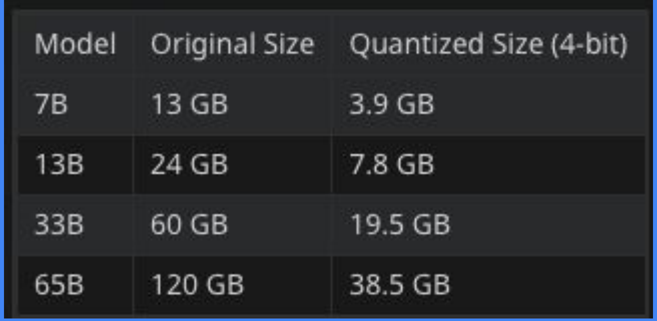


Figure 4: Training of LLAMA 2-CHAT: This process begins with the **pretraining** of LLAMA 2 using publicly available online sources. Following this, we create an initial version of LLAMA 2-CHAT through the application of **supervised fine-tuning**. Subsequently, the model is iteratively refined using Reinforcement Learning with Human Feedback (**RLHF**) methodologies, specifically through rejection sampling and Proximal Policy Optimization (PPO). Throughout the RLHF stage, the accumulation of **iterative reward modeling data** in parallel with model enhancements is crucial to ensure the reward models remain within distribution.

Methodology / Approach

- Set up model(s) locally on machine, created simple server and client to interact with them
- LLama.cpp is a library for performing efficient inference, used python bindings to get a server up and running
- [Demo time]
- Next steps (for building: More advanced server features, hosting it on PC, better client
- Next steps (for evaluation): download more quantized model and benchmark speed (tokens per second)

Results / Quantization



Model	Original Size	Quantized Size (4-bit)
7B	13 GB	3.9 GB
13B	24 GB	7.8 GB
33B	60 GB	19.5 GB
65B	120 GB	38.5 GB

- No results yet regarding testing speeds of quantized models, though key aspects of projects (running the LLMS locally) is working
- What is quantization?
  - To predict tokens, need to load model’s weights into memory (RAM or GPU)
  - Even the smallest 7b parameter LLama 2 model requires 28GB of Ram (more than laptop)
  - By default, these parameters generally are 32-bit or 4 byte floats. 7b *4 byte -> 28 GB
  - Quantization is process of reducing precision of weights, e.g. to 16-bit, 8-bit, 4-bit or lower
  - Model in demo “models/llama-2-7b-chat.Q5_K_M.gguf”:
    - Q5: 5 bit quantization, gguf is model file format for using with llama.cpp library
- Image source, local llama reddit wiki: https://www.reddit.com/r/LocalLLaMA/wiki/index/

Discussion

- Outside of helping the field, valuable experience to finally get myself to get one of these local LLMs running. Had been meaning to for a while. Everyone has had the ChatGPT moment, it is like that but it is on your laptop
- Could be valuable to have data points on how well the llama models run on the devices I am testing (macbook memory now, soon PC GPU)