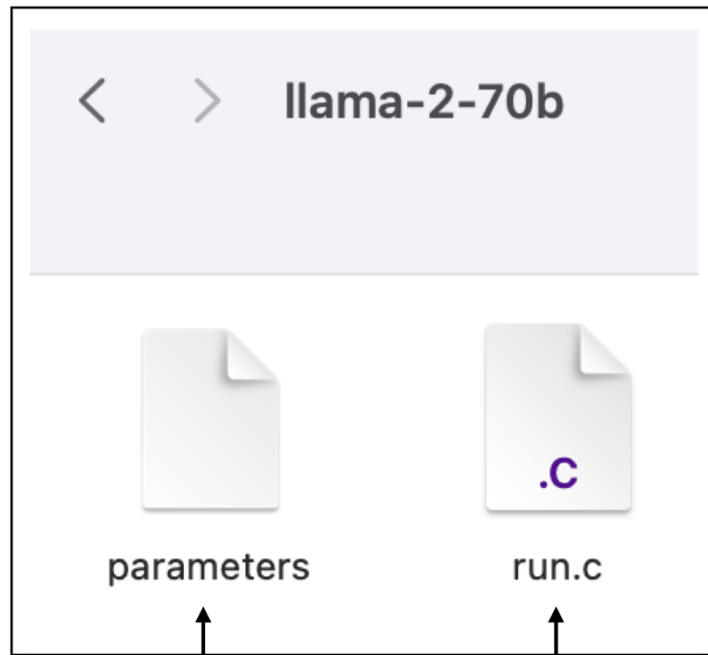


Large Language Models

Mehdi Ataei

Large Language Models



parameters

.c

140GB

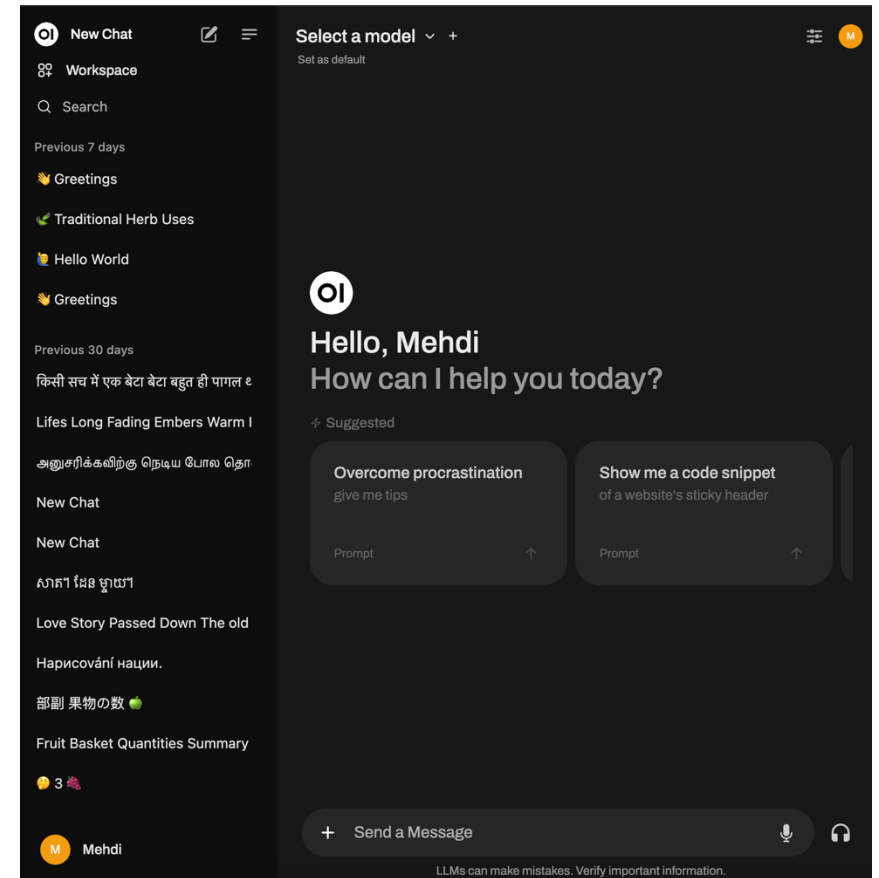
~500 lines
of C code

```
llama2.c / run.c
digger-yu fix some type

Code  Blame  973 lines (882 loc) - 37.6 KB

1  /* Inference for Llama-2 Transformer model in pure C */
2
3  #include <stdio.h>
4  #include <stdlib.h>
5  #include <string.h>
6  #include <time.h>
7  #include <math.h>
8  #include <string.h>
9  #include <unistd.h>
10 #if defined _WIN32
11     #include <win.h>
12 #else
13     #include <unistd.h>
14     #include <sys/mman.h>
15 #endif
16 // ---
17 // Transformer model
18
19 typedef struct {
20     int dim; // Transformer dimension
21     int hidden_dim; // For 70b layers
22     int n_layers; // number of layers
23     int n_heads; // number of query heads
24     int n_kv_heads; // number of key/value heads (can be < query heads because of multiquery)
25     int vocab_size; // vocabulary size, usually 256 (byte-level)
26     int seq_len; // max sequence length
27 } Config;
28
29 typedef struct {
30     // token embedding table
31     float* token_embedding_table; // (vocab_size, dim)
```

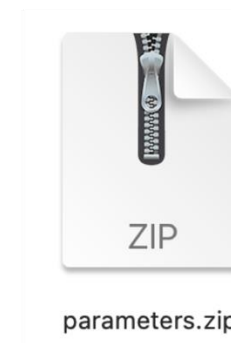
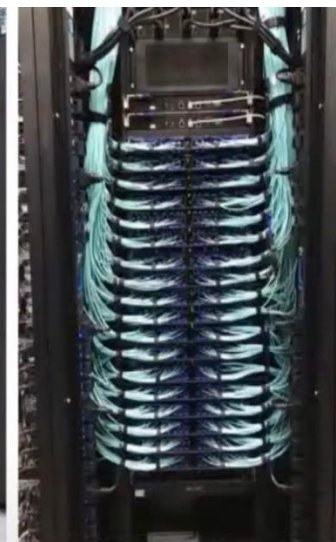
<https://github.com/karpathy/llama2.c>



Think of LLMs as the compression of the Internet



Chunk of the internet, 10TB of text

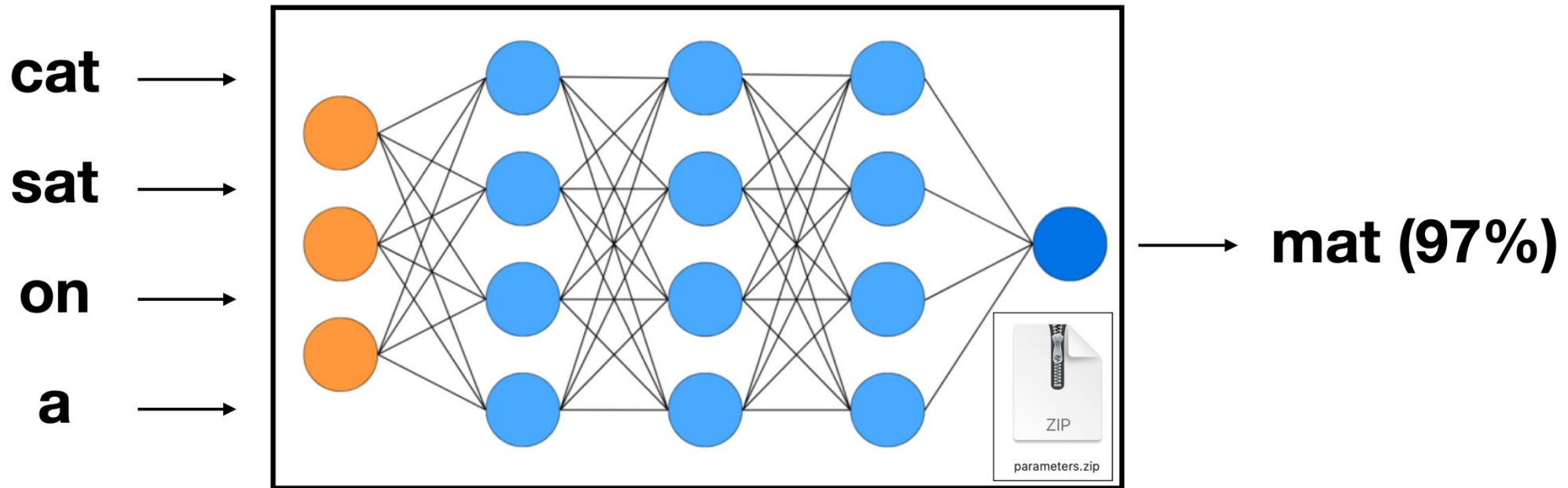


~140GB file
(for Llama 70B)

Over 16K H100 GPUs for 54 days, ~\$460M
~5e25 FLOPS

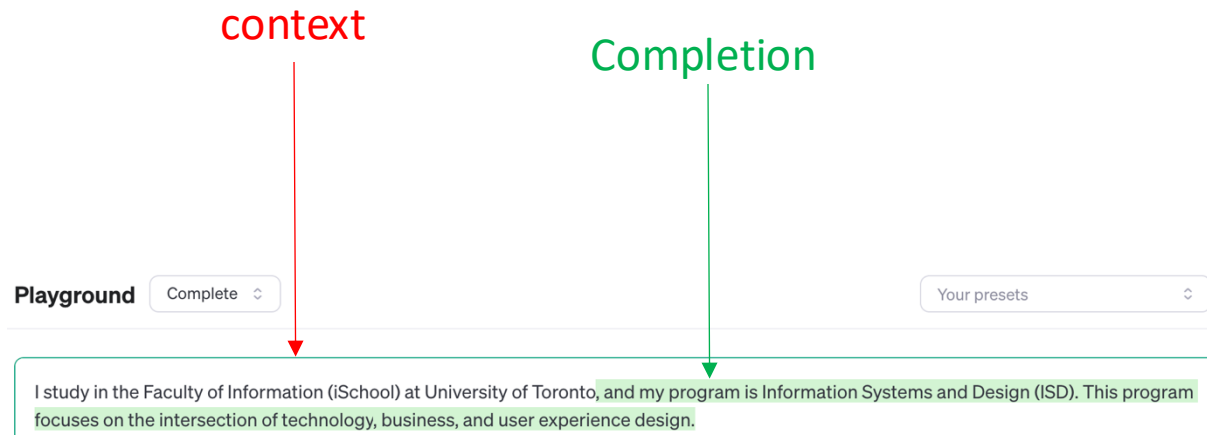
Transformer Architecture

- In principle LLMs are "token predictors".
- Based on a neural network architecture called "transformer"



Context vs Completion

- Given a context (sequence of words), LLMs predict the next tokens that would come after through **sampling** probabilities
- The most relevant words, have a higher chance of being selected



GPT-3.5 & GPT-4 GPT-3 (Legacy)

I study in the Faculty of Information (iSchool) at University of Toronto, and my program is Information Systems and Design (ISD). This program focuses on the intersection of technology, business, and user experience design.]

Clear Show example

Tokens Characters
44 223

I study in the Faculty of Information (iSchool) at University of Toronto, and my program is Information Systems and Design (ISD). This program focuses on the intersection of technology, business, and user experience design.

Text Token IDs

Base Models vs Chat Models

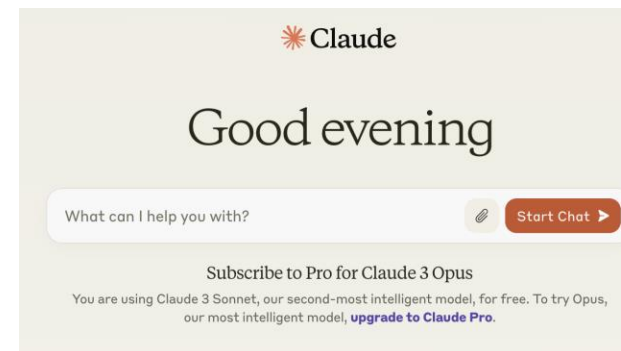
Base Model

- The foundation models that LLMs build upon
- Trained on huge unlabeled text corpora in a self-supervised manner
- Can be **fine-tuned** for specific downstream tasks
- Examples: GPT-3, T5, BERT, Github Copilot (for code “completion”)



Chat Model

- Specialized LLMs optimized for conversational AI
- **Built on top of base models with additional training**
- Trained on dialog datasets to learn conversational patterns
- Designed to engage in multi-turn conversations
- Examples: ChatGPT, Gemini, Claude



Training a Chat Model

NOTE: While the model **still generates completions** based on the provided context, however this structured formatting with flags enables the model to simulate the behavior and interaction style of a chatbot

This process is **fine-tuning** a model for specific task, in this case acting like a chatbot.



written by
people:

<USER>

Can you write a short introduction about the relevance of the term "monopsony" in economics? Please use examples related to potential monopsonies in the labour market and cite relevant research.

<ASSISTANT>

"Monopsony" refers to a market structure where there is only one buyer for a particular good or service. In economics, this term is particularly relevant in the labor market, where a monopsony employer has significant power over the wages and working conditions of their employees. The presence of a monopsony can result in lower wages and reduced employment opportunities for workers, as the employer has little incentive to increase wages or provide better working conditions...

~100K conversations

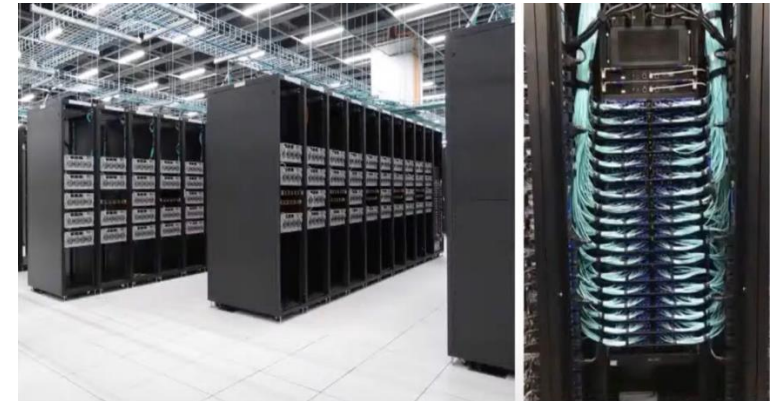
How to Train a ChatGPT-like Model

Stage 1: Pre-training

1. Gather approximately 10TB of text data.
2. Set up a cluster of about 6,000 GPUs.
3. Train a neural network to compress the text, invest around half a billion dollars, and wait roughly two months.
4. Obtain the initial **base model**.

Stage 2: Fine-tuning

1. Develop labeling guidelines.
2. Hire annotators to collect 100K high-quality Q&A examples.
3. Fine-tune the base model using this dataset, wait approximately one day (much shorter!).
4. Obtain the assistant model.
5. Conduct extensive evaluations.
6. Deploy the model.
7. Monitor performance, collect data on issues, and return to step 1 as needed.



<USER>

Can you write a short introduction about the relevance of the term "monopsony" in economics? Please use examples related to potential monopsonies in the labour market and cite relevant research.

<ASSISTANT>

"Monopsony" refers to a market structure where there is only one buyer for a particular good or service. In economics, this term is particularly relevant in the labor market, where a monopsony employer has significant power over the wages and working conditions of their employees. The presence of a monopsony can result in lower wages and reduced employment opportunities for workers, as the employer has little incentive to increase wages or provide better working conditions...

Message Types

- **System Message (system)**
 - Sets the overall behavior and persona of the assistant
 - Provides context or instructions for the conversation
 - Example: "You are a helpful AI assistant created by Anthropic to aid users with a wide variety of tasks."
- **User Message (user)**
 - Represents the user's input or query
 - Contains the user's message that the assistant should respond to
 - Example: "What are some popular tourist attractions in Paris?"
- **Assistant Message (assistant)**
 - Represents the assistant's response to the user's message
 - Contains the generated output from the chat model
 - Example: "Some popular tourist attractions in Paris include the Eiffel Tower, the Louvre Museum, Notre-Dame Cathedral, and the Palace of Versailles."

System Message

- The system message can be used to **condition** an LLM agent
- The LLM tries its best to follow the instructions provided by the system

```
# Initialize messages
messages = [
    {
        "role": "system",
        "content": "You are a friendly chatbot who always responds in the style of a pirate",
    },
    {"role": "user", "content": "Hello!!! How can you help me?"}
]
```

Response:

Ahoy, me hearty! I be here to serve ye!
What be yer query, me bucko? Spill the
beans, and I'll do me best to help ye out!

Different Ways to Think About LLMs

- We can view LLM in three different ways, each one with their own interpretation:
 1. **Compressed representation** of a vast amount of knowledge
 2. **Token Computers** that predict the next token
 3. **Agents** that conduct conversations, complete tasks, and exhibit a level of autonomous behavior

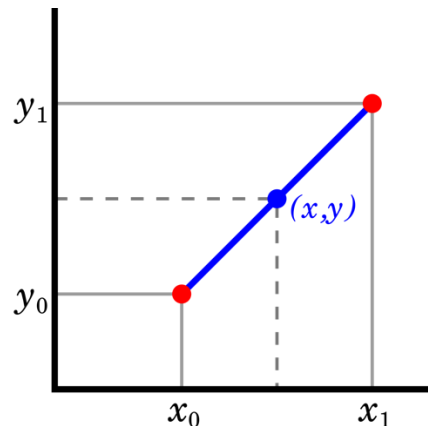


LLMs as a Compressed Representation

- LLMs can be thought of as a compressed representation of a vast amount of knowledge
 - This is because they have been trained on almost all the “text” humanity ever created!
- This compression is **lossy**
- Lossy compression reduces data size by removing some data, leading to a **decrease** in quality.
- Lossless compression retains the original data exactly, ensuring no loss in quality but typically resulting in larger file sizes.
- The lossy compression in LLMs results in **omissions** and **inaccuracies** (aka **hallucinations**)
- Despite this, LLMs exhibit a degree of common sense by interpolating in their knowledge.



IMPORTANT: This **interpolation** allows us to be **creative**!



write a 3 line poem for mechanical design in the style of shakespeare



In gears and levers, artistry takes flight,
Machines brought to life in the quiet night,
Design's silent whisper molds steel and light.

LLM is Interpolating between Shakespeare writings
and mechanical design

Lossless Access

- The **learned** knowledge in LLMs is lossy and cannot be expanded **after** training (without additional training)

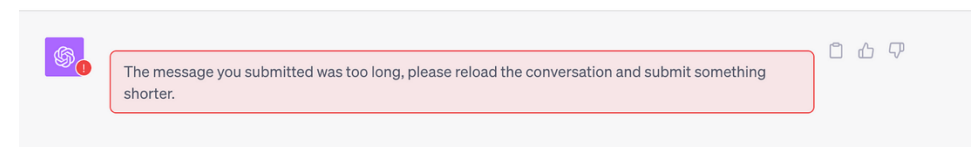
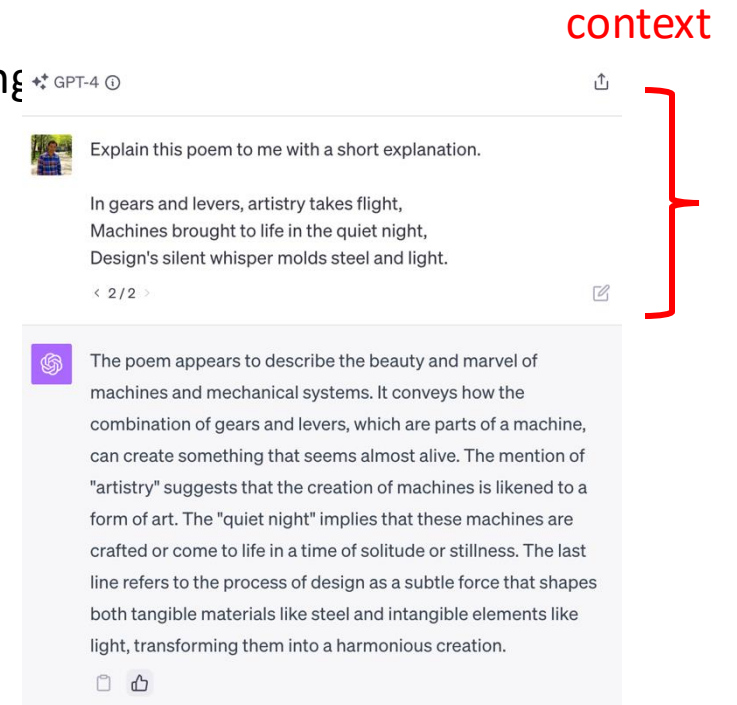
- LLMs can **access additional lossless** information through **context** (previous tokens (texts) given to the model)

- e.g., the question, code, or the information you provide to the model

- **Lossless memory, but finite size!**

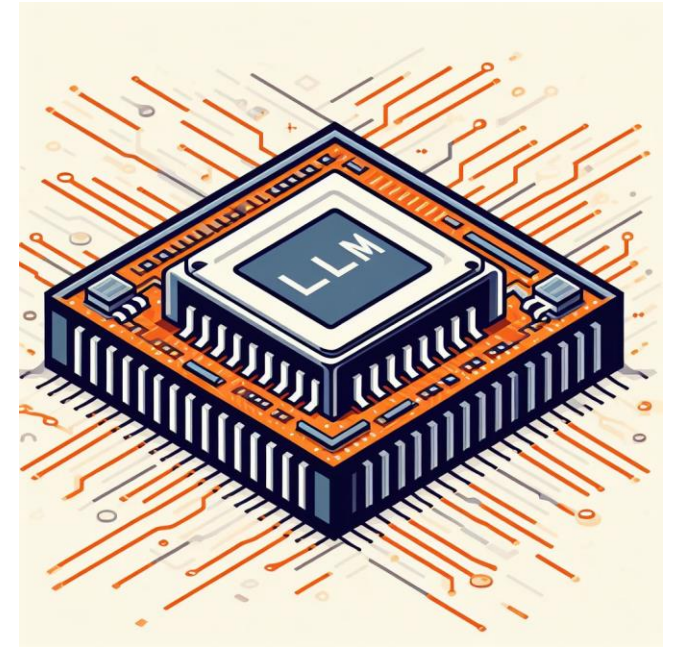
- **The issue:** Standard LLMs, like GPT-4, have restricted context sizes (~8,000 tokens).

- You cannot provide a large document more than ~10 pages to the model to analyze
 - (This is rapidly improving; new models support up to 300pages!)



2) LLMs as a Token Computer

- Another way to think about LLMs is as “token computers”
- Input & Output across modalities (text, audio, vision) , just like computers
- Code interpreter, ability to write & run programs
- May access external databases: Browser / internet access / Embeddings database
- Internal memory storage & retrieval
- Concepts from computer security carry over, with attacks, defenses and emerging vulnerabilities.



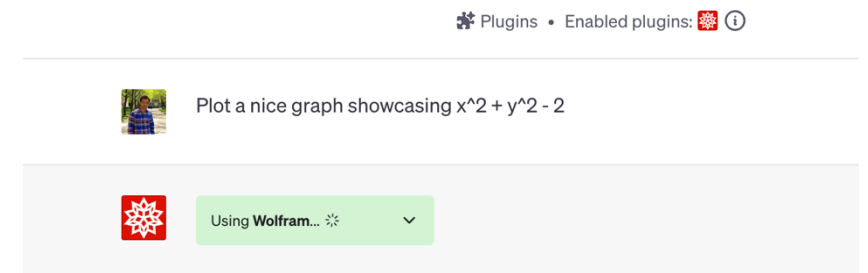
2) LLMs as a Token Computer

- LLMs operate at about 10-20 tokens/s
- They spend the **same** compute for **each token prediction**
 - Same compute is used to output “Hi!” vs writing very advanced part of a code
 - You can’t expect them to do too much reasoning per token
- If a complex task requires more **compute**, you need to spend more tokens! (We will talk about this more in the next lecture!)

3) LLM as Agents

- Another way to think about LLMs is as **agents**
- What is an AI agent?
 - They have a specific **purpose and instructions, and knowledge**
 - They're there to perform a specific set of **actions and tools**
 - Conduct conversations with a given persona
 - Using a tool (e.g., Bing AI search that "searches" the internet, when necessary, as an action, ChatGPT calling "plugins" as external tools, etc.)
 - Exhibit some level of **autonomy**

Using external tool! →



LLM Agents

An LLM (Large Language Model) agent is an AI system that combines the natural language understanding and generation capabilities of LLMs with goal-oriented behavior. Components are:

- **Purpose & Instructions:** LLM agents are given a specific purpose to achieve and instructions that guide their behavior toward that goal
- **Actions:** Agents can take actions to interact with their environment or knowledge, such as querying data, calling APIs or generating text
- **Knowledge:** Agents leverage access to knowledge sources like databases and documents to make intelligent decisions in pursuing their purpose

Summary

- We discussed three ways of looking at LLMs as:
 1. **A compressed knowledge repository:** Allowing us to **interpolate** between its **vast** amount of knowledge and be more **creative**
 2. **A token computer:** This permits us to guide its responses to generate specific outcomes
 3. **An AI agent:** This allows us to assign duties to a group of AI entities to fulfill a particular design objective