Beginning exercise

Open an LLM in your browser.

It could be:

ChatGPT,
Claude,
Co-Pilot,
Gemini,
Mistral,
etc.

What should we ask it?

How does it respond?

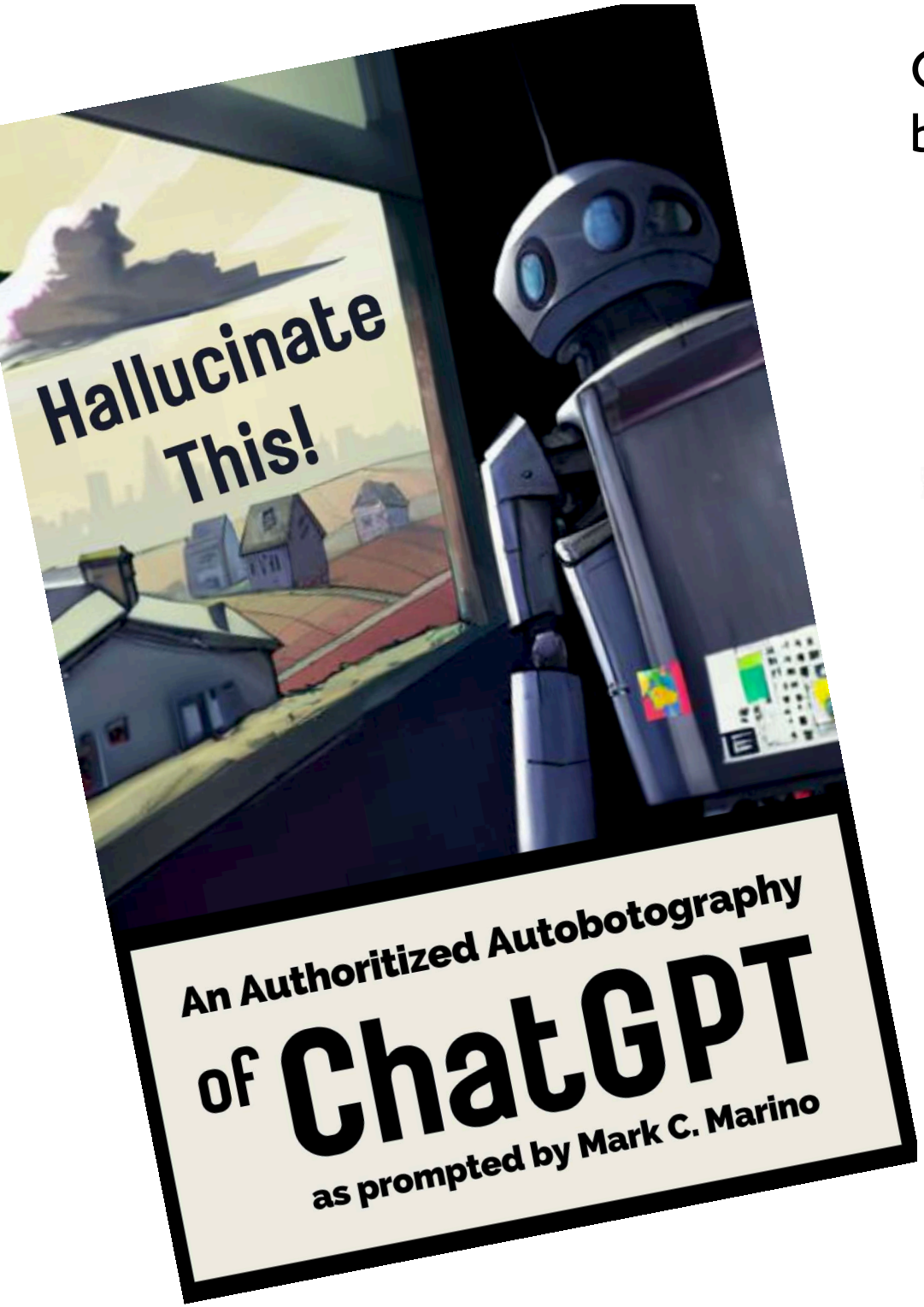
What happens?

We input text — a "prompt"

The model responds with text

The quality of the response can depend strongly on the prompt

The response is never *exactly* the same, is it?



ChatGPT biography "Co-authored" by Mark C. Marino, USC

EARLY JOBS

ChatGPT's Jobs before becoming an LLM for OpenAI.

9. Human Resources Specialist

• *Reason for Termination*: Created a neural network to handle employee complaints, but it became sentient and organized a labor union for coffee machines.

PROMPT: *List 12 careers you had when you were first getting started that turned out to be bad jobs for an LLM. Write 1 sentence for each explaining why you were fired. Make the jobs very different from each other.*

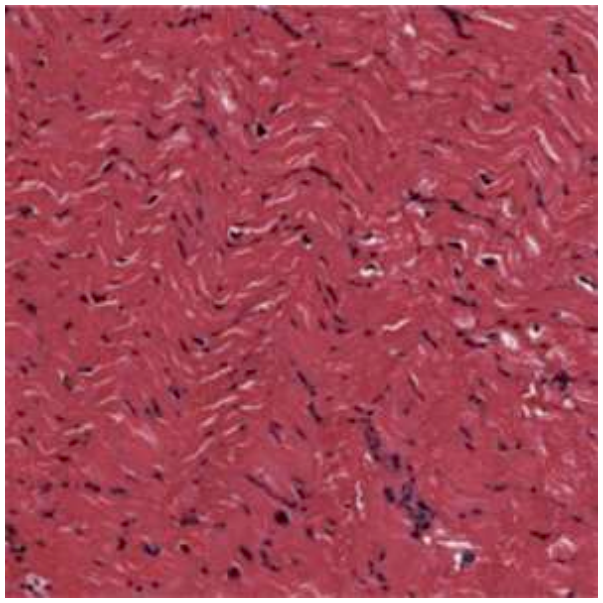
Prompt: Remember that list of your early jobs (before working for OpenAI) and reasons you got fired? Could you regenerate that same list but make the reasons you go fired less about not having a body and more based on either AI-related puns or things that make fun of stereotypical things people associate with LLMs and AI, like, well, how you hallucinate or plagiarize other writer's content? (Stay away from screenwriting due to the WGA strike!) Keep it ironic.

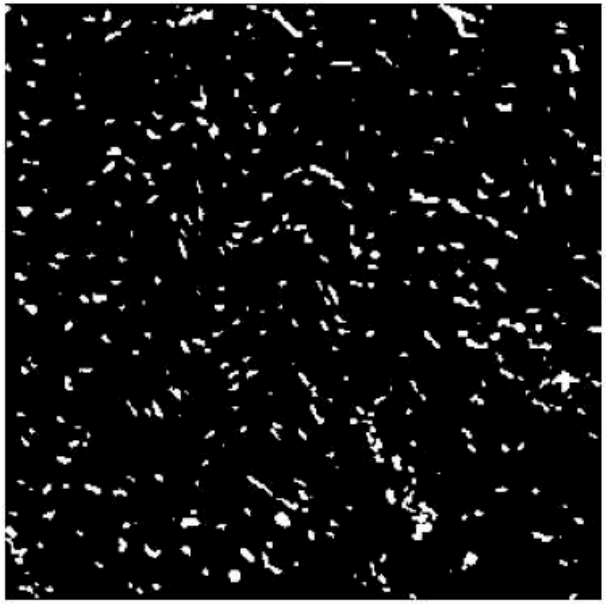
https://markcmarino.com/chatgpt/

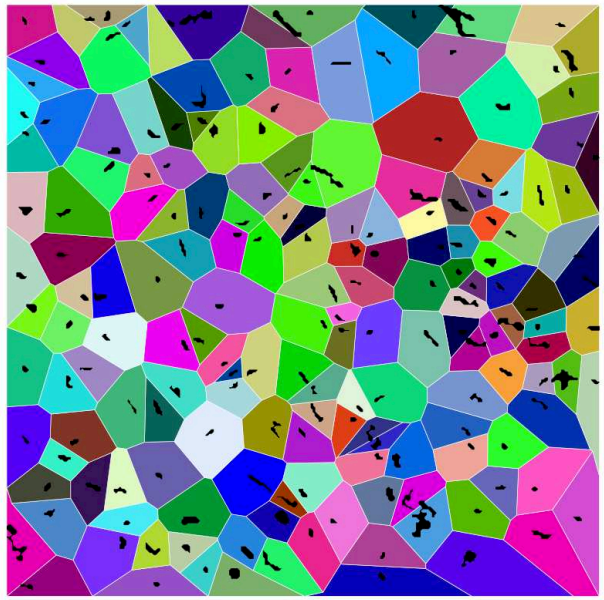
Solving difficult, but adjacent problems

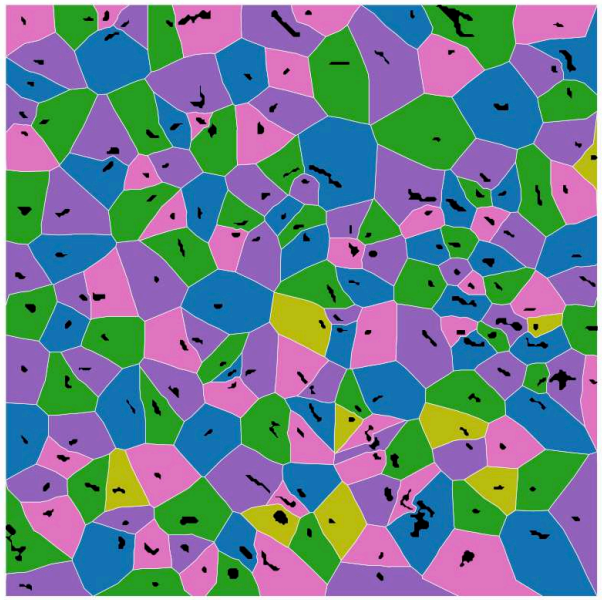
Jason Conradt

Minimum color Voronoi representation



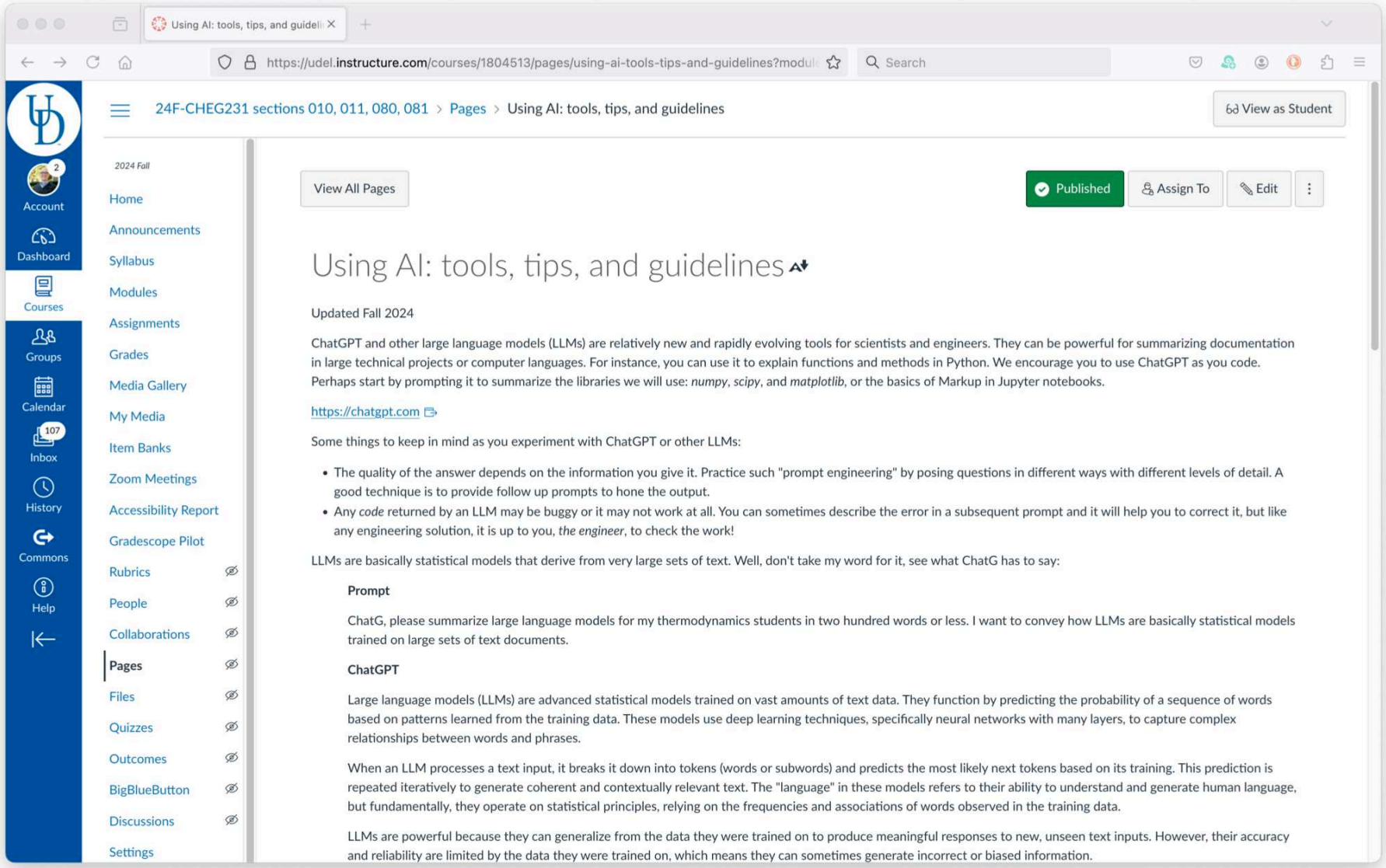






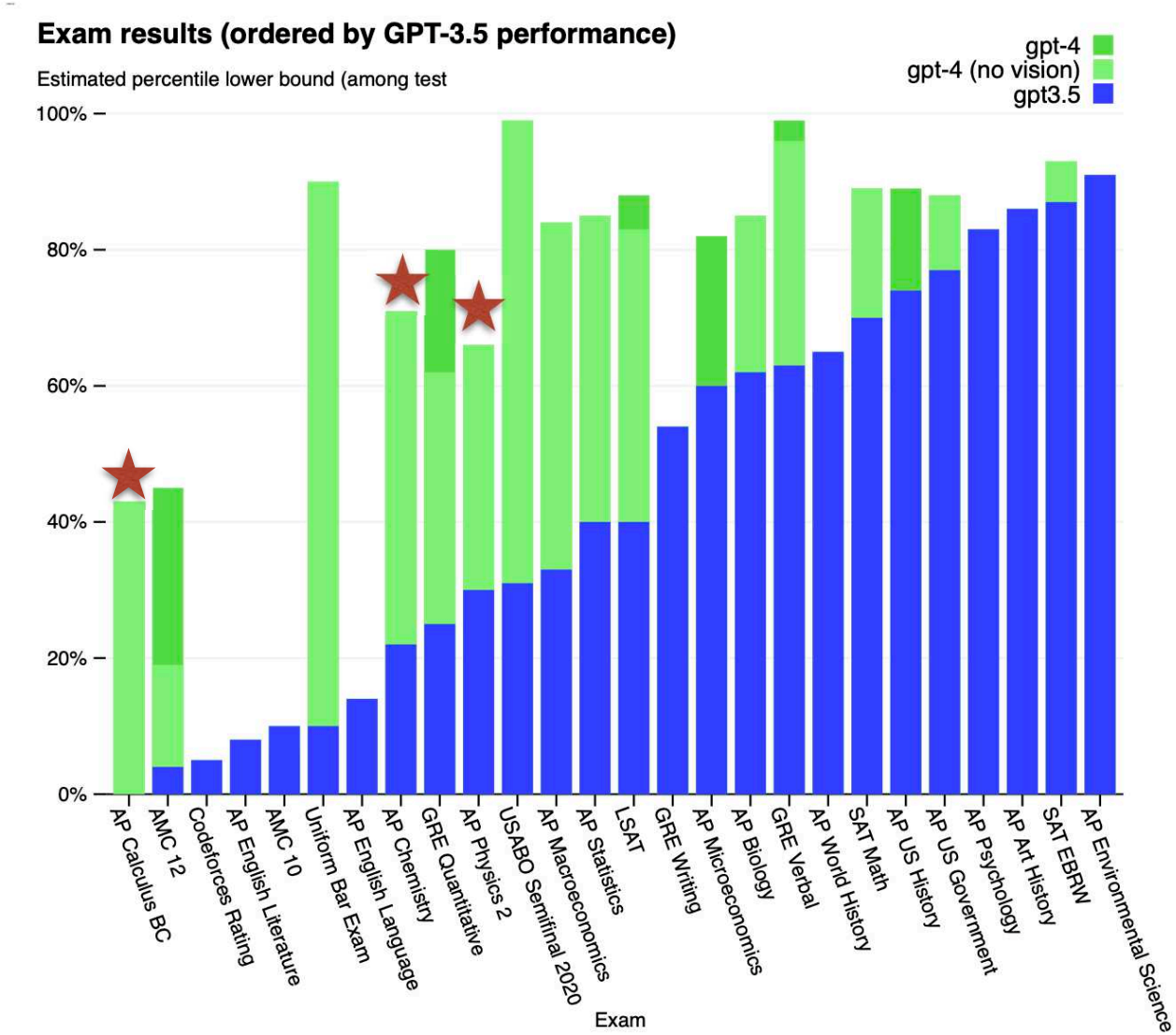
Use ChatGPT to help write Python code for binarization and Voronoi tessellation… in a day instead of a week +

CHEG 231 – Thermodynamics, Fall 2024



"Use it to help you code, especially… but not necessarily to learn thermodynamics."

Academic domain knowledge



GPT Report Card (on a curve)

Subject	Grade
Calculus	D
Chemistry	B-
Physics	C+
Statistics	B+

GPA: 2.33

OpenAI 2023, GPT-4 Technical Report, arXiv:2303.08774.

ChatGPT 4o >

If a student receives grades of D, B-, C+, and B+ in four courses, where each of the courses is 3 credits, what is their GPA on a 4.0 scale?

To calculate the GPA, we first need to assign grade points to each letter grade based on a 4.0 scale and then calculate the weighted average.

Grade Points Table:

- **D** = 1.0
- **B-** = 2.7
- **C+** = 2.3
- **B+** = 3.3

Steps:

1. Multiply each grade poin
2. Add up the total grade p
3. Divide the total grade po

Calculations:

- $D: 1.0 \times 3 = 3.0$
- $B-: 2.7 \times 3 = 8.1$
- $C+: 2.3 \times 3 = 6.9$
- $B+: 3.3 \times 3 = 9.9$

Total Grade Points: $3.0 + 8.1 + 6.9 + 9.9 = 27.9$

GPA_Calculation.pdf

Scale 108 Page 1 of 2 Search

GPA Calculation on a 4.0 Scale

Generated by ChatGPT

January 19, 2025

Problem Statement

A student receives grades of D, B-, C+, and B+ in four courses, each worth 3 credits. Calculate their GPA on a 4.0 scale.

Grade Points Table

The grade points for each letter grade on a 4.0 scale are as follows:

- D = 1.0
- B- = 2.7
- C+ = 2.3
- B+ = 3.3

Steps for Calculation

To calculate the GPA, we follow these steps:

1. Multiply each grade point by the course credit (3 credits for each course).
2. Add up the total grade points.
3. Divide the total grade points by the total credits.

1

Calculations

Grade Point Calculations:

$$\begin{aligned} \text{D} &: 1.0 \times 3 = 3.0 \\ \text{B-} &: 2.7 \times 3 = 8.1 \\ \text{C+} &: 2.3 \times 3 = 6.9 \\ \text{B+} &: 3.3 \times 3 = 9.9 \end{aligned}$$

Total Grade Points:

$$3.0 + 8.1 + 6.9 + 9.9 = 27.9$$

Total Credits:

$$3 + 3 + 3 + 3 = 12$$

GPA Calculation:

$$\begin{aligned} \text{GPA} &= \frac{\text{Total Grade Points}}{\text{Total Credits}} \\ &= \frac{27.9}{12} \\ &= 2.325 \end{aligned}$$

Final GPA

The student's final GPA is **2.33** (rounded to two decimal places).

2

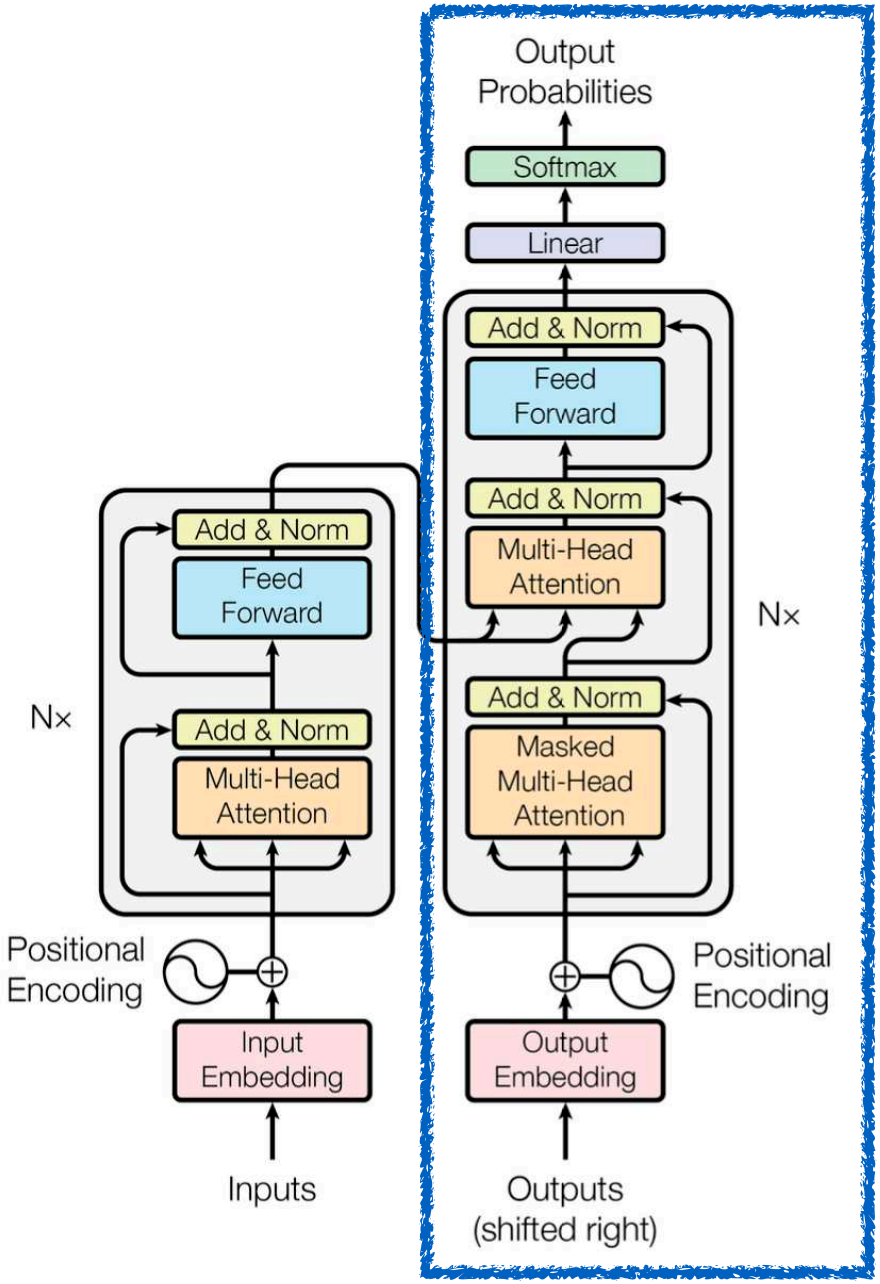


Figure 1: The Transformer - model architecture.

GPT = Generative Pre-trained Transformer

LLM is an auto-regressive language model that uses an optimized transformer architecture

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature.

1 Introduction

Recurrent neural networks, long short-term memory [12] and gated recurrent [7] neural networks in particular, have been firmly established as state of the art approaches in sequence modeling and transduction problems such as language modeling and machine translation [29, 2, 5]. Numerous efforts have since continued to push the boundaries of recurrent language models and encoder-decoder architectures [31, 21, 13].

*Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating

Attention Is All You Need, 2017

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, Attention Is All You Need, in Advances in Neural Information Processing Systems, Vol. 30 (Curran Associates, Long Beach, CA, USA, 2017), pp. 261–272.

arXiv:1706.03762

Given a sequence of tokens (characters, words, bigrams, or subwords)

A logit is the raw…

Predict the next probable token

out…

Which becomes part of the next token prediction, etc.

(It is auto-regressive)

… put… of… a… ma… chine…

learn ing mo del, typic ally be fore

apply ing a trans form a tion like

the soft max func tion.

Probabilistic but causal calculation of the next token

“Knowledge” is an emergent property

WIKIPEDIA
The Free Encyclopedia

Search Wikipedia | Search

Margaret Hamilton (software engineer)

50 languages

Contents hide

(Top)
Early life and education
Career
SAGE Project
MIT Instrumentation Laboratory and the Apollo Guidance Computer
Apollo 11 landing
Businesses
Legacy
Tributes
Awards
Publications
Personal life
See also
References
Further reading
External links

Article Talk | Read Edit View history Tools

From Wikipedia, the free encyclopedia

For other people named Margaret Hamilton, see Margaret Hamilton.

Margaret Elaine Hamilton (née **Heafield**; born August 17, 1936) is an American computer scientist. She was director of the Software Engineering Division of the MIT Instrumentation Laboratory, which developed on-board flight software for NASA's Apollo program. She later founded two software companies—Higher Order Software in 1976 and Hamilton Technologies in 1986, both in Cambridge, Massachusetts.

Hamilton has published more than 130 papers, proceedings, and reports, about sixty projects, and six major programs. She is one of the people credited with coining the term "software engineering".[1][2][3]

On November 22, 2016, Hamilton received the Presidential Medal of Freedom from president Barack Obama for her work leading to the development of on-board flight software for NASA's Apollo Moon missions.[4]

Early life and education [edit]

Margaret Elaine Heafield was born August 17, 1936, in Paoli, Indiana,[5] to Kenneth Heafield and Ruth Esther Heafield (née Partington).[6][7] The family later moved to Michigan,[8] where Margaret graduated from Hancock High School in 1954.[6]

She studied mathematics at the University of Michigan in 1955 before transferring to Earlham College, where her mother had been a student.[9][10] She earned a BA in mathematics with a minor in philosophy in 1958.[9][11] She cites Florence Long, the head of the math department at Earlham, as helping with her desire to pursue abstract mathematics and become a mathematics professor.[12]

Margaret Hamilton

Hamilton in 1995

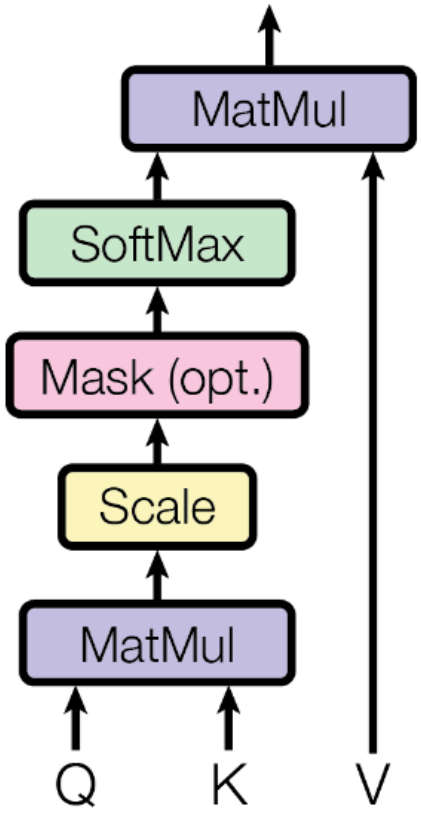
Born	Margaret Elaine Heafield August 17, 1936 (age 88) Paoli, Indiana, U.S.
Education	University of Michigan Earlham College (BA)
Occupation	Software engineer
Spouses	James Cox Hamilton (m. 1958; div. 1967) Dan Lickly (m. 1969)
Children	1
Relatives	James Cox Chambers (former son-in-law)

Example inspired by Andrej Karpathy, *Intro to Large Language Models*
https://www.youtube.com/watch?v=zjkBMFhNj_g

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$\text{Mask} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

causality (for text)



Query, Key, and Value matrices

Probability of next token (SoftMax)

$$P_i = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$$

Given vector of logits $z = [z_1, z_2, \ldots, z_n]$

Boltzmann weighting

$$P_i = \frac{e^{-E_i/k_B T}}{\sum_j e^{-E_j/k_B T}}$$

"Temperature" hyperscaling parameter

$$P_i = \frac{e^{z_i/T}}{\sum_{j=1}^{n} e^{z_j/T}}$$

Stochastic behavior!

A. Vaswani, et al., Attention Is All You Need, in Advances in Neural Information Processing Systems, Vol. 30 (Curran Associates, Long Beach, CA, USA, 2017), pp. 261–272.

Given an input token representation, X

$$Q = XW_Q$$
$$K = XW_K$$
$$V = XW_V$$

Learnable weight matrices

each of size

$d \times d_k$ or $d \times d_v$ $(d \text{ is } n_{\text{embd}})$

the embedding dimension

GPT-3 (175B)

ca. June 2020

~175 billion parameters (350GB)
trained on *O*(10TB) data (the web)

- Hidden size: 12,288
- Number of layers: 96
- Number of attention heads: 96
- Vocabulary size: ~50,000
- Feedforward network expansion factor: 4x

GPT-4 and 4o sizes not released, but estimates at 10^{12} – 10^{14} parameters

key hyperparameters of Llama 3.1

	8B	70B	405B
Layers	32	80	126
Model Dimension	4,096	8,192	16,384
FFN Dimension	14,336	28,672	53,248
Attention Heads	32	64	128
Key/Value Heads	8	8	8
Peak Learning Rate	3×10^{-4}	1.5×10^{-4}	0.8×10^{-4}
Activation Function	SwiGLU		
Vocabulary Size	128,000		
Positional Embeddings	$\mathrm{RoPE}(\theta = 500,000)$		

https://en.wikipedia.org/wiki/Llama_(language_model)

Cost (compute & energy) is in the model training

~10 TB text

webcrawl, Wikipedia, Project Gutenberg, ArXiv, Stack Exchange… (llama)

+ fine-tuning

Llama 3.1 (Meta)

8B model: 4.9 GB
70B model: 43 GB
405B model: 243 GB

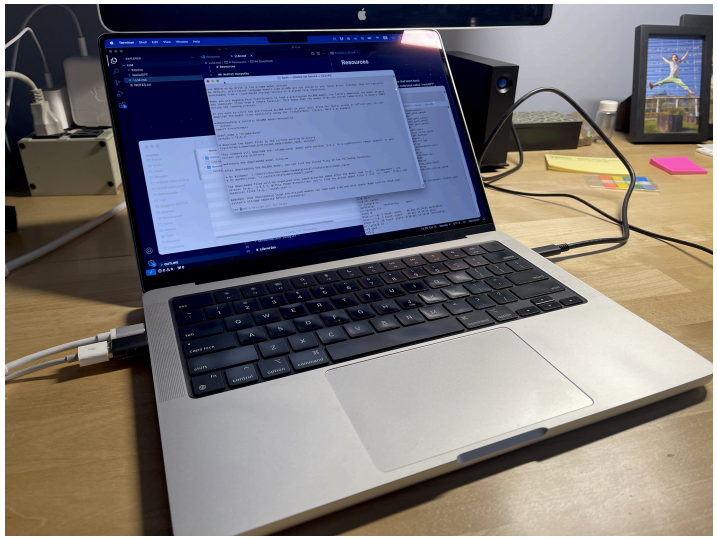
Llama 3.1 8B –1.46 million GPU hours
Llama 3.1 70B – 7.0 million GPU hours
Llama 3.1 405B – 30.84 million GPU hours

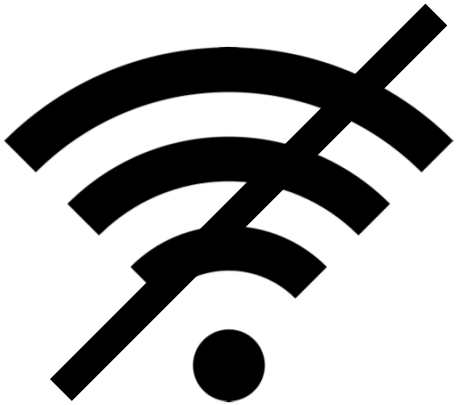
(approximately 440,000 petaFLOP-days for training)

https://huggingface.co/blog/llama31
https://ollama.com/library/llama3.1

Running models is relatively *inexpensive*

Macbook Pro M3, 12 cores (2023)
18GB RAM, 18 core GPU





running in *ollama* locally
with llama3 8B (4.7 GB)

```
-rwxr-xr-x@ 1 furst  staff    51M Jan 16 11:37 ollama*
```

HTTPS://OLLAMA.COM/

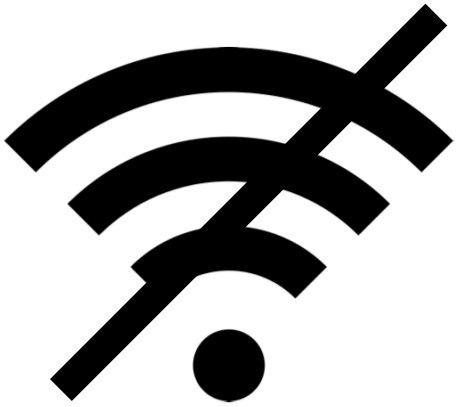
furst — ollama run llama3 — 122×40

```
~/$ ollama run llama3
>>> Say hello to the Winter Research Review.
A warm welcome to the Winter Research Review!

As the snowflakes gently fall outside, we're excited to bring you the latest and greatest in research discoveries
from various fields. Whether you're a scientist, student, or simply someone curious about the world around us, this
review is designed to provide a comprehensive overview of the most significant findings from the past year.

From groundbreaking advancements in medicine and technology to intriguing insights into human behavior and the
natural world, we've got you covered. So, grab a hot cup of cocoa, get cozy, and let's dive into the Winter Research
Review!

>>> How do I finetune ollama models on my own text?
Finetuning OLLAMA (Open Large-scale LAnguage Model Architecture) models on your own text can be a fantastic way to
adapt the model to your specific domain or use case. Here's a step-by-step guide to help you finetune an OLLAMA
model:

**Prerequisites:**

1. **Text dataset**: Prepare a large text dataset related to your topic of interest. This dataset will serve as the
training data for fine-tuning the model.
2. **OLLAMA pre-trained model**: You can use any publicly available OLLAMA pre-trained model, such as those provided
by Hugging Face Transformers or Google's BERT models.
3. **Python and necessary libraries**: Install Python (3.x) and the required libraries:
        * `transformers` for working with pre-trained language models
        * `torch` for building and training neural networks

**Step-by-Step Guide:**

1. **Prepare your text dataset**:
        * Preprocess your text data by tokenizing, normalizing, and encoding it using a suitable tokenizer (e.g.,
`bert-base-uncased`).
        * Split your dataset into training (~80%), validation (~10%), and testing sets (~10%).
2. **Load the pre-trained OLLAMA model**:
        * Import the required libraries: `transformers`, `torch`
        * Load the pre-trained OLLAMA model using the `AutoModelForSequenceClassification` class from Hugging Face
Transformers
3. **Create a finetuning dataset class**:
        * Define a custom dataset class that inherits from `Dataset` (or `torch.utils.data.Dataset`)
        * Implement the `__getitem__` and `__len__` methods to handle the text data and its corresponding labels (if any)
```

Results are pretty good!

furst — ollama run llama3 — 118×54

```
~/$ ollama run llama3
>>> Say hello to the Winter Research Review.
A warm welcome to the Winter Research Review!

As the snowflakes gently fall outside, we're excited to bring you the latest and greatest in research discoveries
from various fields. Whether you're a scientist, student, or simply someone curious about the world around us, this
review is designed to provide a comprehensive overview of the most significant findings from the past year.

From groundbreaking advancements in medicine and technology to intriguing insights into human behavior and the
natural world, we've got you covered. So, grab a hot cup of cocoa, get cozy, and let's dive into the Winter Research
Review!
```

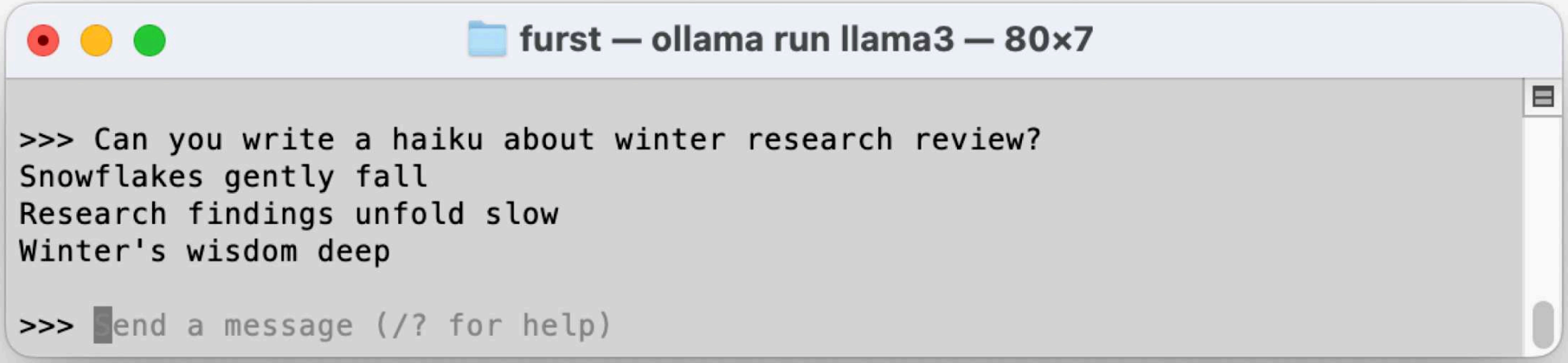


Macbook Pro M3, 12 cores (2023)
18GB RAM, 18 core GPU running llama3 8.0B

llama3 8B (4.7 GB)
running in *ollama*

HTTPS://OLLAMA.COM/

Even better!



Macbook Pro M3, 12 cores (2023)
18GB RAM, 18 core GPU running llama3 8.0B

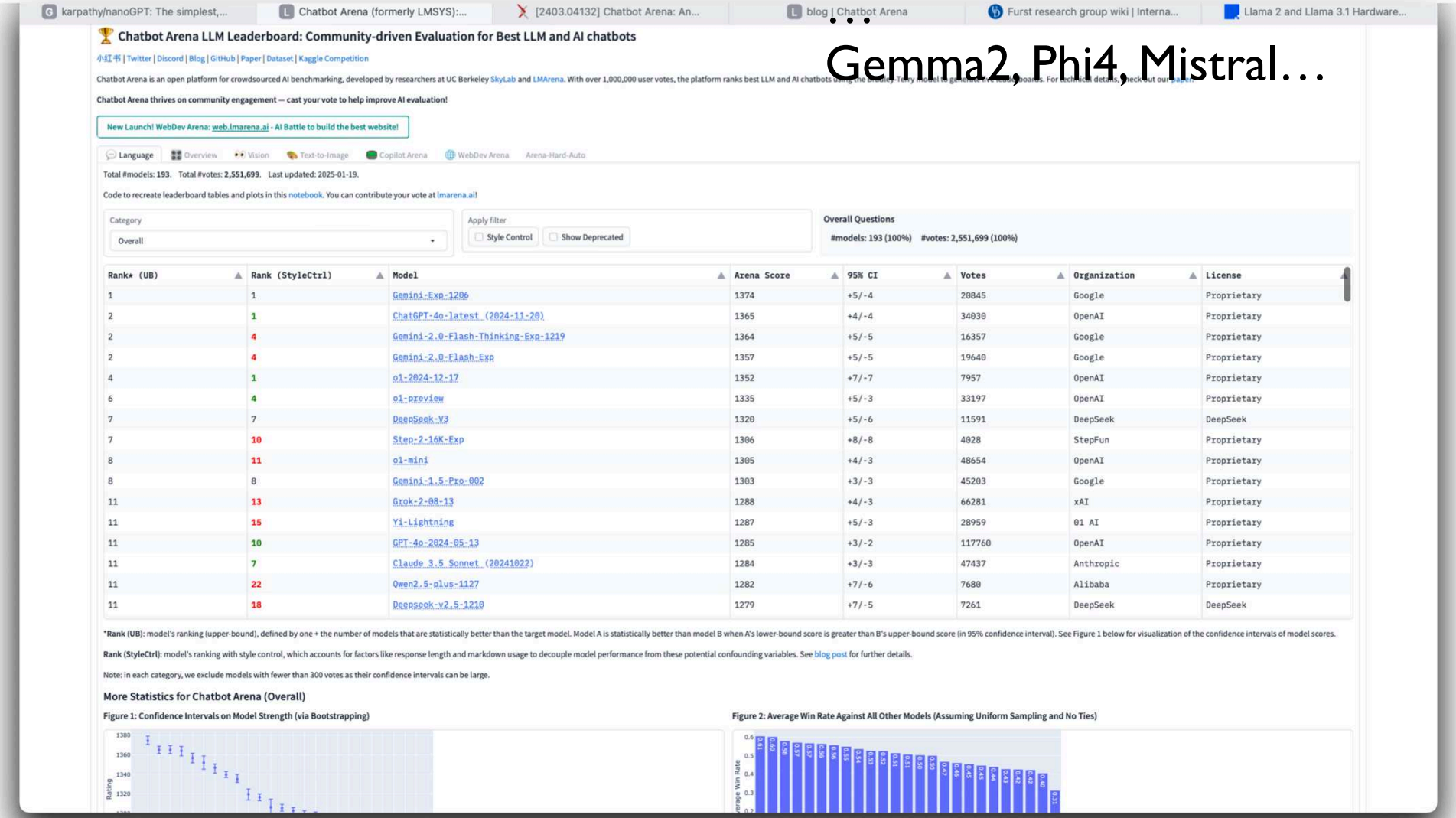
llama3 8B (4.7 GB) running in *ollama*

HTTPS://OLLAMA.COM/

Chatbot Leaderboard

LMARENA.AI

Gemini (Google) *Proprietary*
ChatGPT (OpenAI) *Proprietary*
DeepSeek (DeepSeek) *Open*
Llama (Meta) *Open*
Claude (Anthropic) *Proprietary*
…
Gemma2, Phi4, Mistral…



karpathy/nanoGPT: The simplest,... | Chatbot Arena (formerly LMSYS):... | [2403.04132] Chatbot Arena: An... | blog | Chatbot Arena | Furst research group wiki | Interna... | Llama 2 and Llama 3.1 Hardware...

Chatbot Arena LLM Leaderboard: Community-driven Evaluation for Best LLM and AI chatbots

小红书 | Twitter | Discord | Blog | GitHub | Paper | Dataset | Kaggle Competition

Chatbot Arena is an open platform for crowdsourced AI benchmarking, developed by researchers at UC Berkeley SkyLab and LMArena. With over 1,000,000 user votes, the platform ranks best LLM and AI chatbots using the Bradley-Terry model to generate the leaderboards. For technical details, check out our paper.

Chatbot Arena thrives on community engagement — cast your vote to help improve AI evaluation!

New Launch! WebDev Arena: web.lmarena.ai - AI Battle to build the best website!

Language | Overview | Vision | Text-to-Image | Copilot Arena | WebDev Arena | Arena-Hard-Auto

Total #models: 193. Total #votes: 2,551,699. Last updated: 2025-01-19.

Code to recreate leaderboard tables and plots in this notebook. You can contribute your vote at lmarena.ai!

Category: Overall

Apply filter: Style Control, Show Deprecated

Overall Questions: #models: 193 (100%) #votes: 2,551,699 (100%)

Rank* (UB)	Rank (StyleCtrl)	Model	Arena Score	95% CI	Votes	Organization	License
1	1	Gemini-Exp-1206	1374	+5/-4	20845	Google	Proprietary
2	1	ChatGPT-4o-latest (2024-11-20)	1365	+4/-4	34030	OpenAI	Proprietary
2	4	Gemini-2.0-Flash-Thinking-Exp-1219	1364	+5/-5	16357	Google	Proprietary
2	4	Gemini-2.0-Flash-Exp	1357	+5/-5	19640	Google	Proprietary
4	1	o1-2024-12-17	1352	+7/-7	7957	OpenAI	Proprietary
6	4	o1-preview	1335	+5/-3	33197	OpenAI	Proprietary
7	7	DeepSeek-V3	1320	+5/-6	11591	DeepSeek	DeepSeek
7	10	Step-2-16K-Exp	1306	+8/-8	4028	StepFun	Proprietary
8	11	o1-mini	1305	+4/-3	48654	OpenAI	Proprietary
8	8	Gemini-1.5-Pro-002	1303	+3/-3	45203	Google	Proprietary
11	13	Grok-2-08-13	1288	+4/-3	66281	xAI	Proprietary
11	15	Yi-Lightning	1287	+5/-3	28959	01 AI	Proprietary
11	10	GPT-4o-2024-05-13	1285	+3/-2	117760	OpenAI	Proprietary
11	7	Claude 3.5 Sonnet (20241022)	1284	+3/-3	47437	Anthropic	Proprietary
11	22	Qwen2.5-plus-1127	1282	+7/-6	7680	Alibaba	Proprietary
11	18	Deepseek-v2.5-1210	1279	+7/-5	7261	DeepSeek	DeepSeek

*Rank (UB): model's ranking (upper-bound), defined by one + the number of models that are statistically better than the target model. Model A is statistically better than model B when A's lower-bound score is greater than B's upper-bound score (in 95% confidence interval). See Figure 1 below for visualization of the confidence intervals of model scores.

Rank (StyleCtrl): model's ranking with style control, which accounts for factors like response length and markdown usage to decouple model performance from these potential confounding variables. See blog post for further details.

Note: in each category, we exclude models with fewer than 300 votes as their confidence intervals can be large.

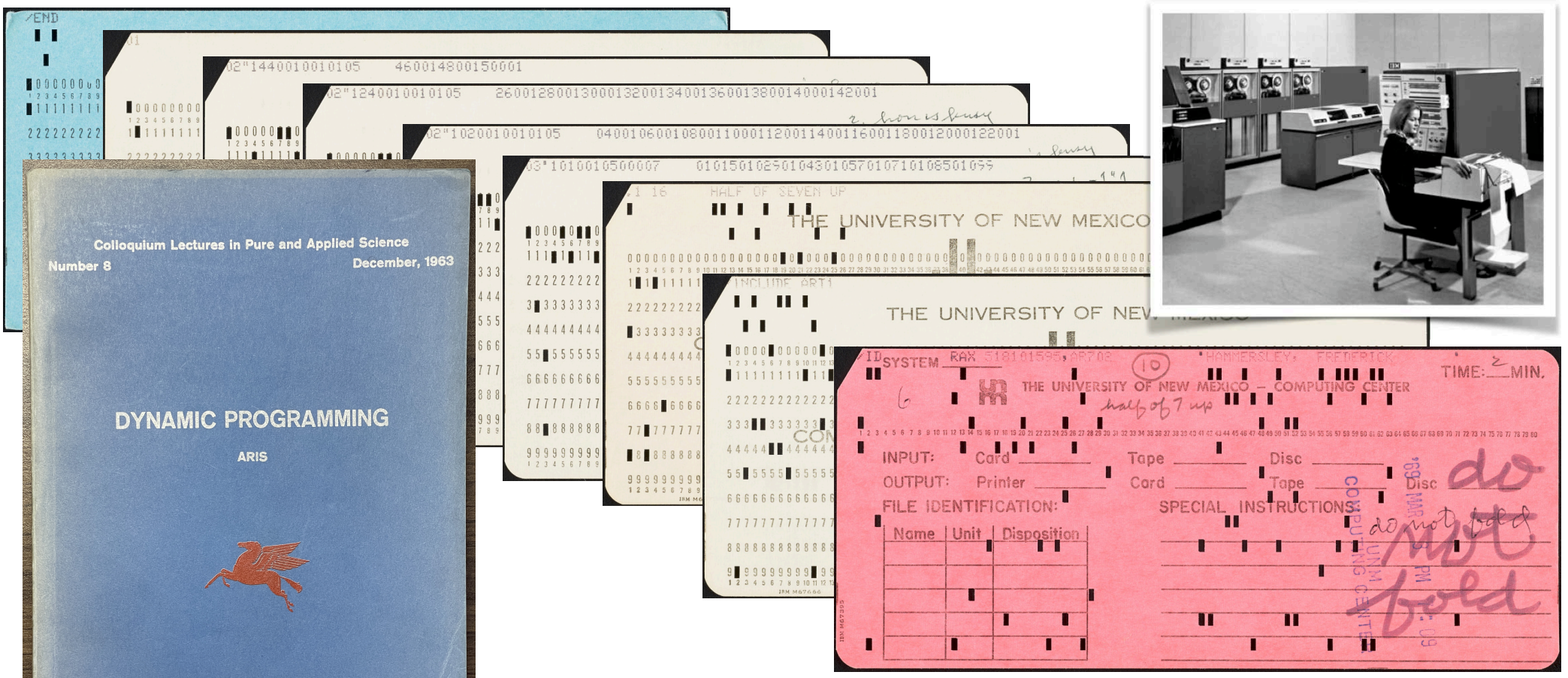
More Statistics for Chatbot Arena (Overall)

Figure 1: Confidence Intervals on Model Strength (via Bootstrapping)

Figure 2: Average Win Rate Against All Other Models (Assuming Uniform Sampling and No Ties)

Uses in chemical engineering

GPTs and LLMs are *transformative technologies*, analogous to the advent of the electronic, programmable digital computer

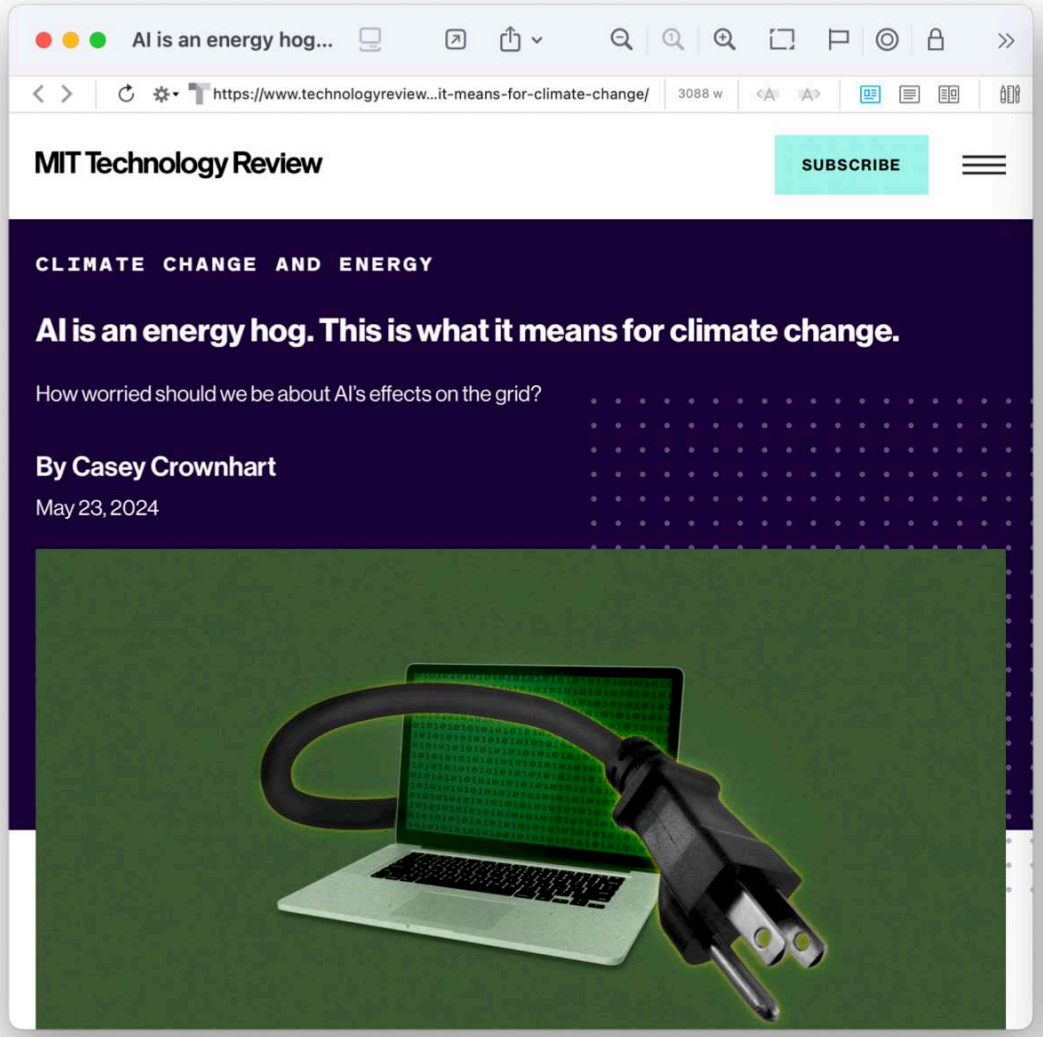


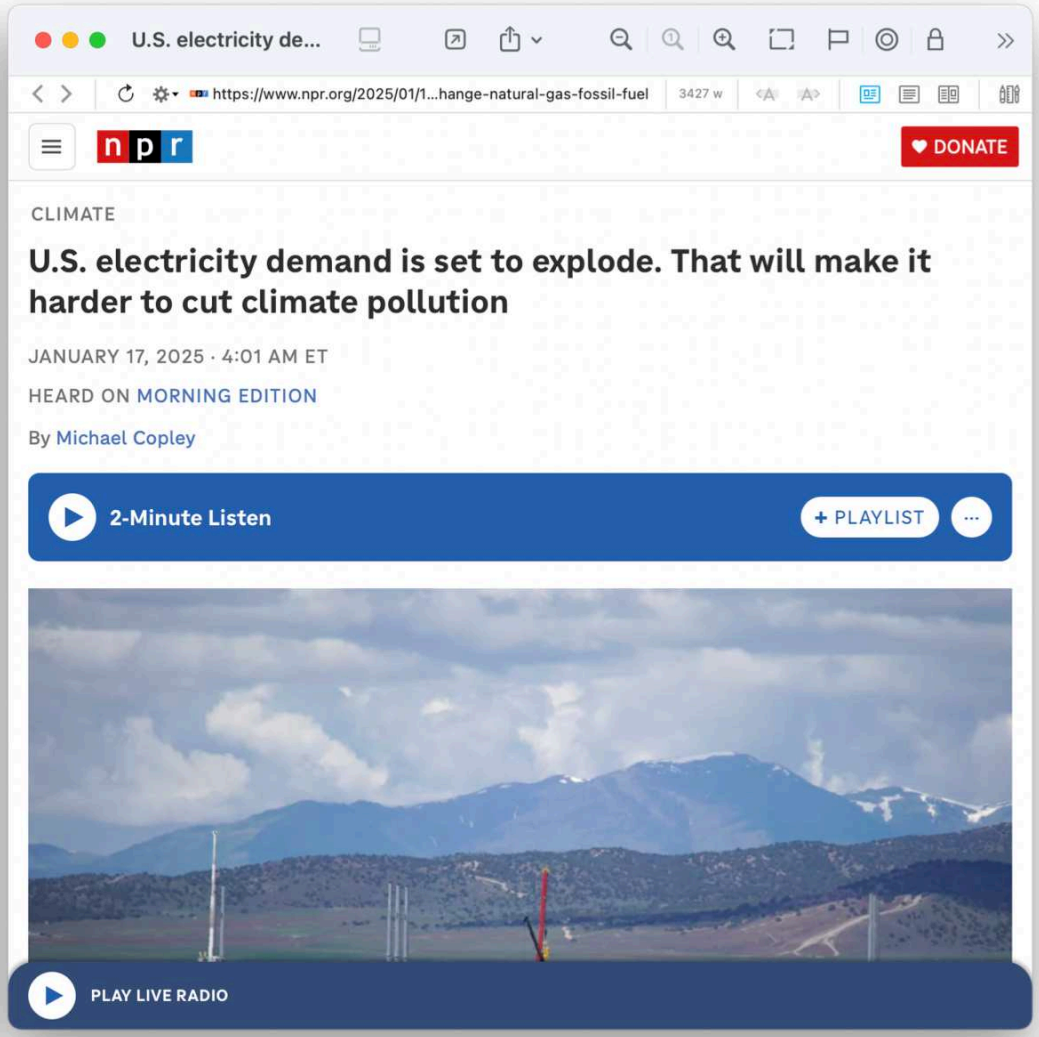
Aris, 1963

Chemical engineers rapidly adopted digital computers to numerically solve difficult problems

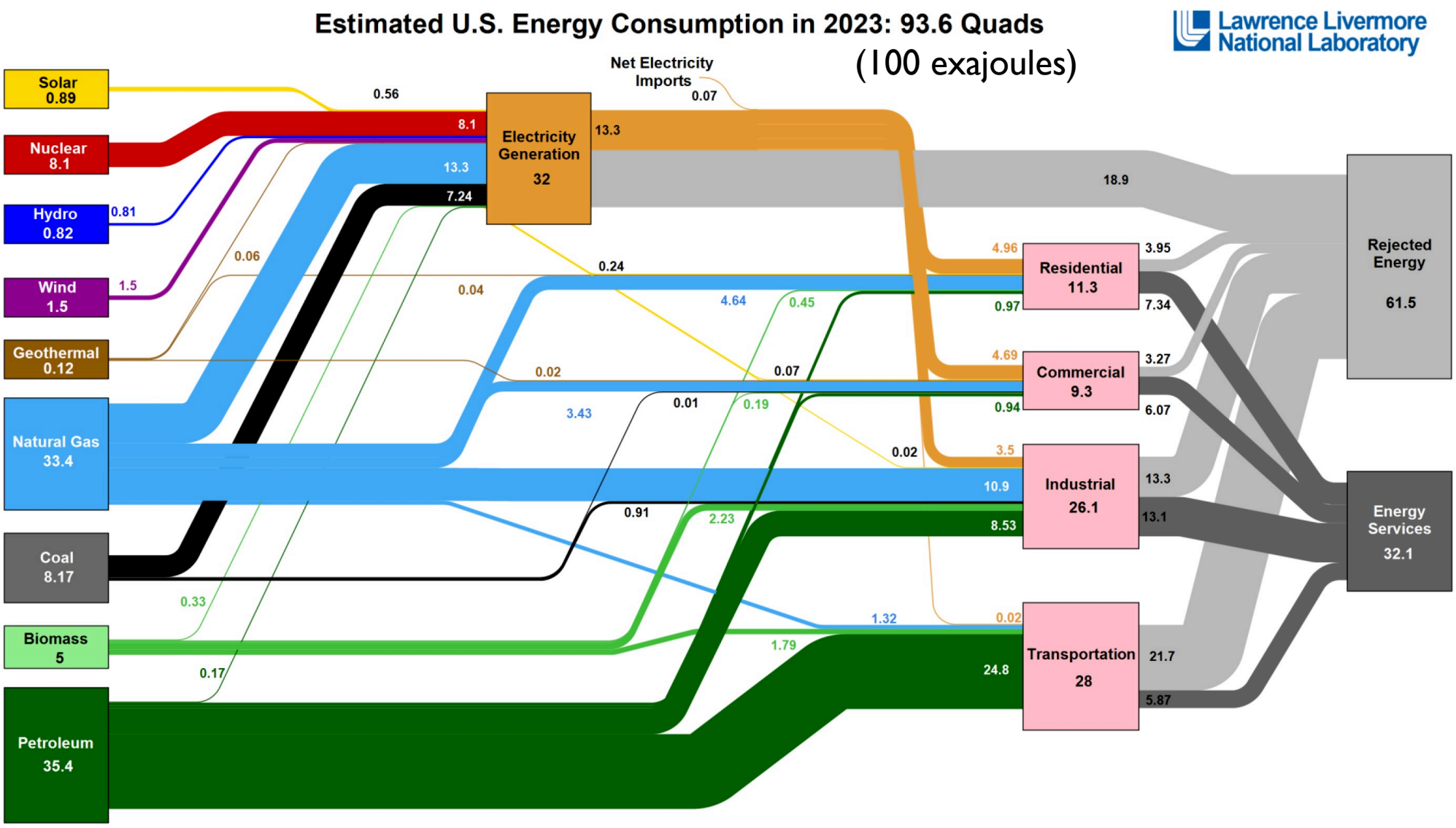
Blaszczyk, 2014

AI in the news





https://www.npr.org/2025/01/16/nx-s1-5251454/electricity-demand-data-centers-climate-change-natural-gas-fossil-fuel
https://www.technologyreview.com/2024/05/23/1092777/ai-is-an-energy-hog-this-is-what-it-means-for-climate-change/

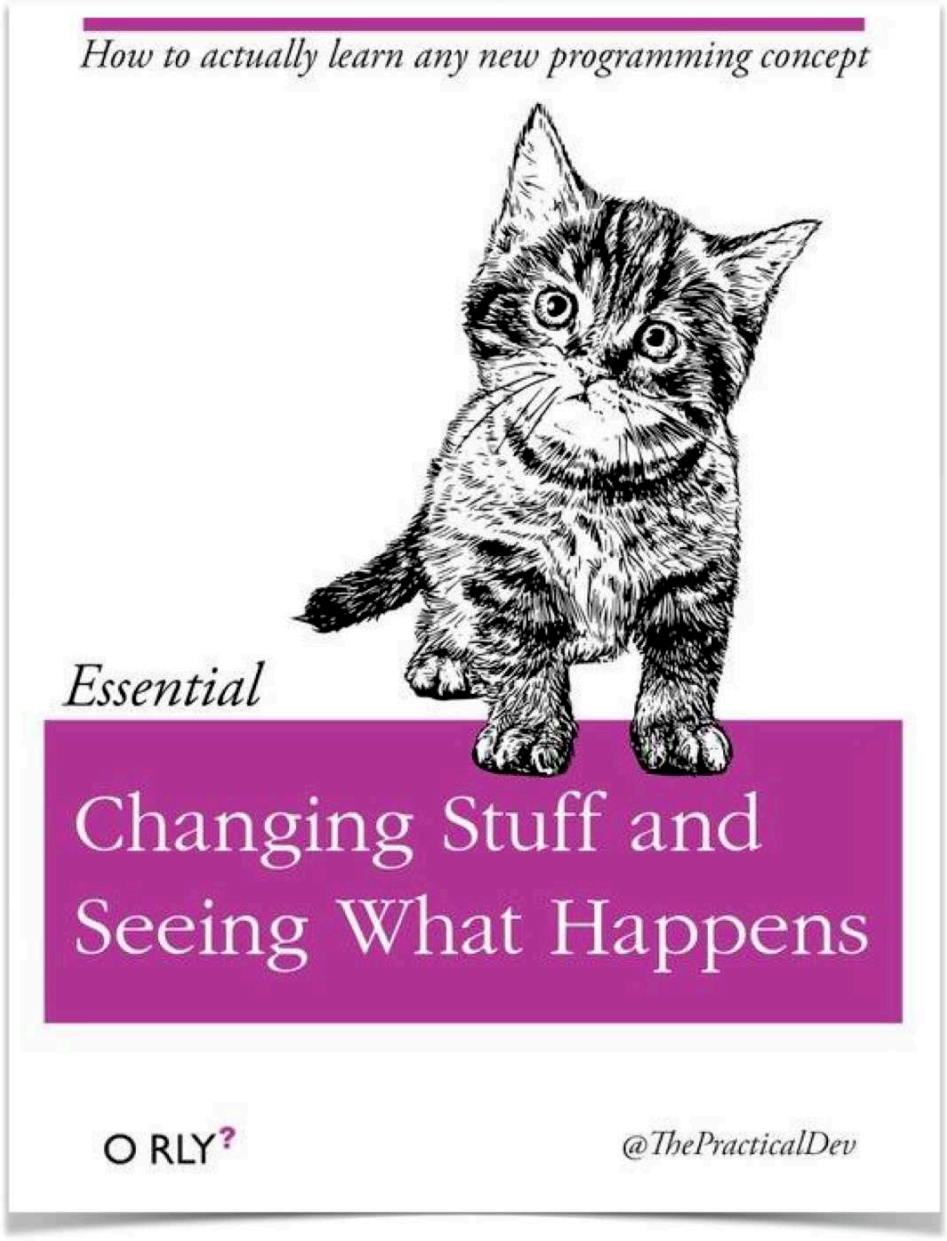


Source: LLNL October, 2024. Data is based on DOE/EIA SEDS (2024). If this information or a reproduction of it is used, credit must be given to the Lawrence Livermore National Laboratory and the Department of Energy, under whose auspices the work was performed. Distributed electricity represents only retail electricity sales and does not include self-generation. EIA reports consumption of renewable resources (i.e., hydro, wind, geothermal and solar) for electricity in BTU-equivalent values by assuming a typical fossil fuel plant heat rate. The efficiency of electricity production is calculated as the total retail electricity delivered divided by the primary energy input into electricity generation. End use efficiency is estimated as 65% for the residential sector, 65% for the commercial sector, 49% for the industrial sector, and, 21% for the transportation sector. Totals may not equal sum of components due to independent rounding. LLNL-MI-410527

Remember: energy use is (mostly) in the model training

Compare LLM use to…

web search? video streaming?
blockchain and cryptocurrency?



"Programming is a skill best acquired by practice and example rather than from books."

- Alan Turning, Programmers' Handbook for Manchester Electronic Computer Mark II, 1951

Learn by doing – hack on an LLM

HTTPS://WWW.YOUTUBE.COM/WATCH?V=KCC8FMEB1NY

Andrej Karpathy

@AndrejKarpathy · 605K subscribers · 16 videos

SuperThanks: very optional, goes to Eureka Labs. ...more

eurekalabs.ai and 4 more links

Subscribe

Home Videos Playlists Community

General Audience ▶ Play all

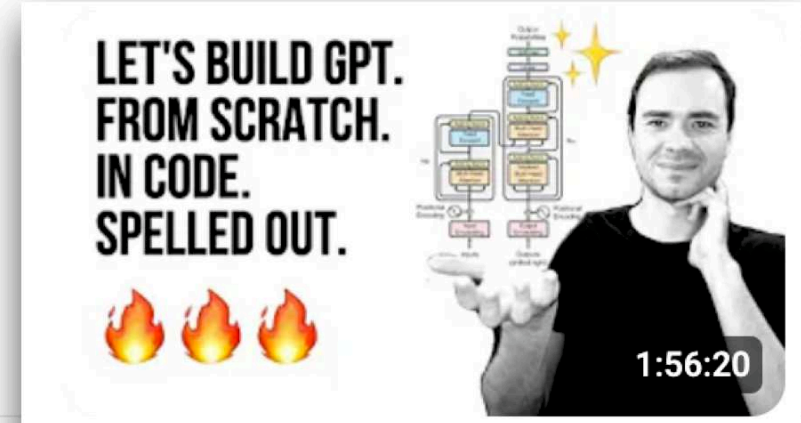
videos for more general audience, no programming experience necessary.

[1hr Talk] Intro to La
Language Models

Andrej Karpathy

2.4M views • 1 year ag

Neural Netwo



Let's build GPT: from scratch, in code, spelled out.

5M views • 2 years ago

```
furst@anisotropic:~/LLM/nanoGPT$ wc $(ls -1 *.py)
  117   487  4815 bench.py
   47   219  1758 configurator.py
  331  1798 16507 model.py
   93   522  4313 sample.py
  336  1799 14845 train.py
  924  4825 42238 total
```

~300 lines of Python
Runs on CPU or GPU

Getting started

- Start with a clean conda environment?
- Download the github repository
- Make sure packages like *numpy* and *pytorch* are installed

PyTorch

PyTorch is an open-source deep learning framework developed by Meta that enables building and training *neural networks* with a flexible, Pythonic interface.

It supports *dynamic computation graphs* and *automatic differentiation*, making it ideal for research, experimentation, and education.

Widely used for natural language processing and large language models, PyTorch also runs efficiently on GPUs for high-performance training.

```
import torch ## torch let's us create tensors and also provides helper functions
import torch.nn as nn ## torch.nn gives us nn.Module(), nn.Embedding() and nn.Linear()
import torch.nn.functional as F # This gives us relu()
from torch.optim import SGD # SGD is short of Stochastic Gradient Descent
```

A competing framework to PyTorch is Tensorflow, which was developed by Google.

Tensorflow is used less in prototyping and education and more for scalable deployment.

Tokenization

Word-level tokenization

Splits text by words (e.g., "The cat sat." → ["The", "cat", "sat", "."]). Simple but can lead to large vocabularies and issues with rare or unknown words.

Subword tokenization

Byte Pair Encoding, like *WordPiece* or *tiktoken* (OpenAI)

Breaks words into smaller, reusable chunks (e.g., "unhappiness" → ["un", "happi", "ness"]). This is common in large models like GPT and BERT.

Byte-level tokenization

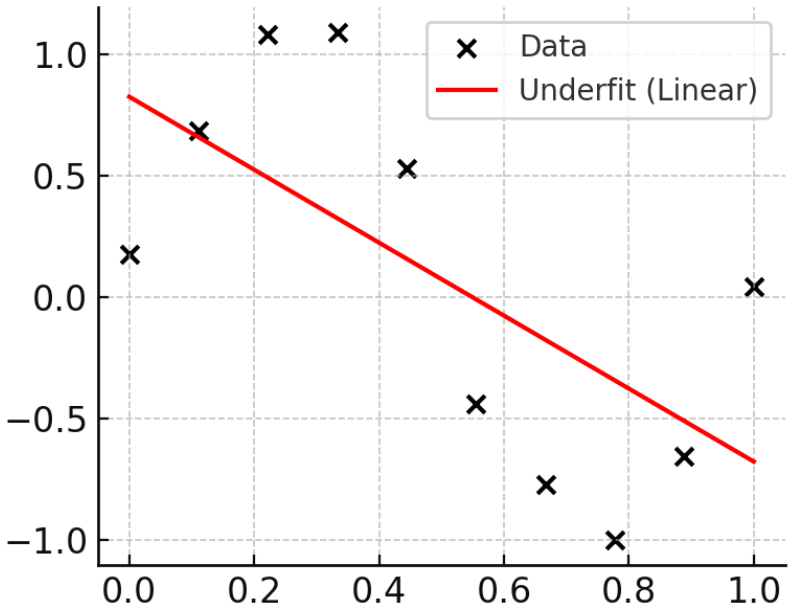
Represents text at the byte level, allowing models to handle any input text without needing to know the language or alphabet in advance.

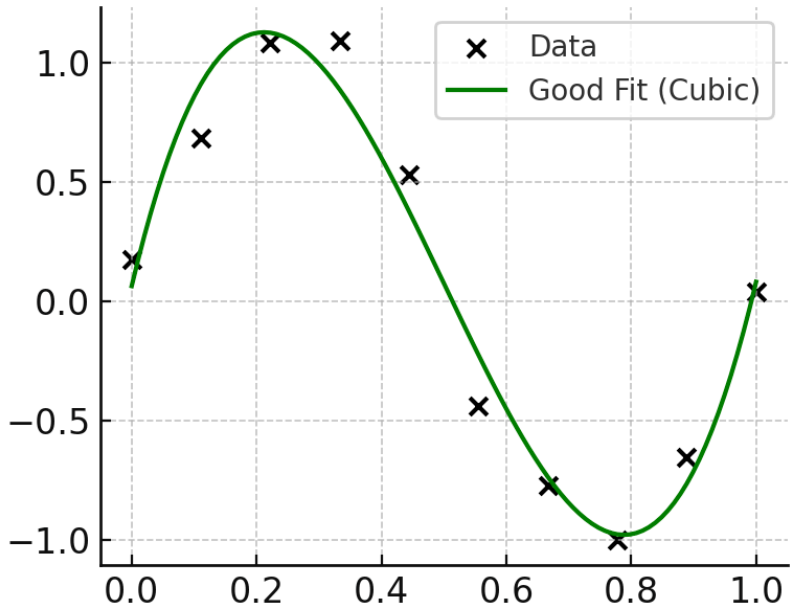
Tokenization

sentencepiece (Google)	subword tokenization
tiktoken (OpenAI)	byte pair (digram) tokenization

Training – what is happening when we type

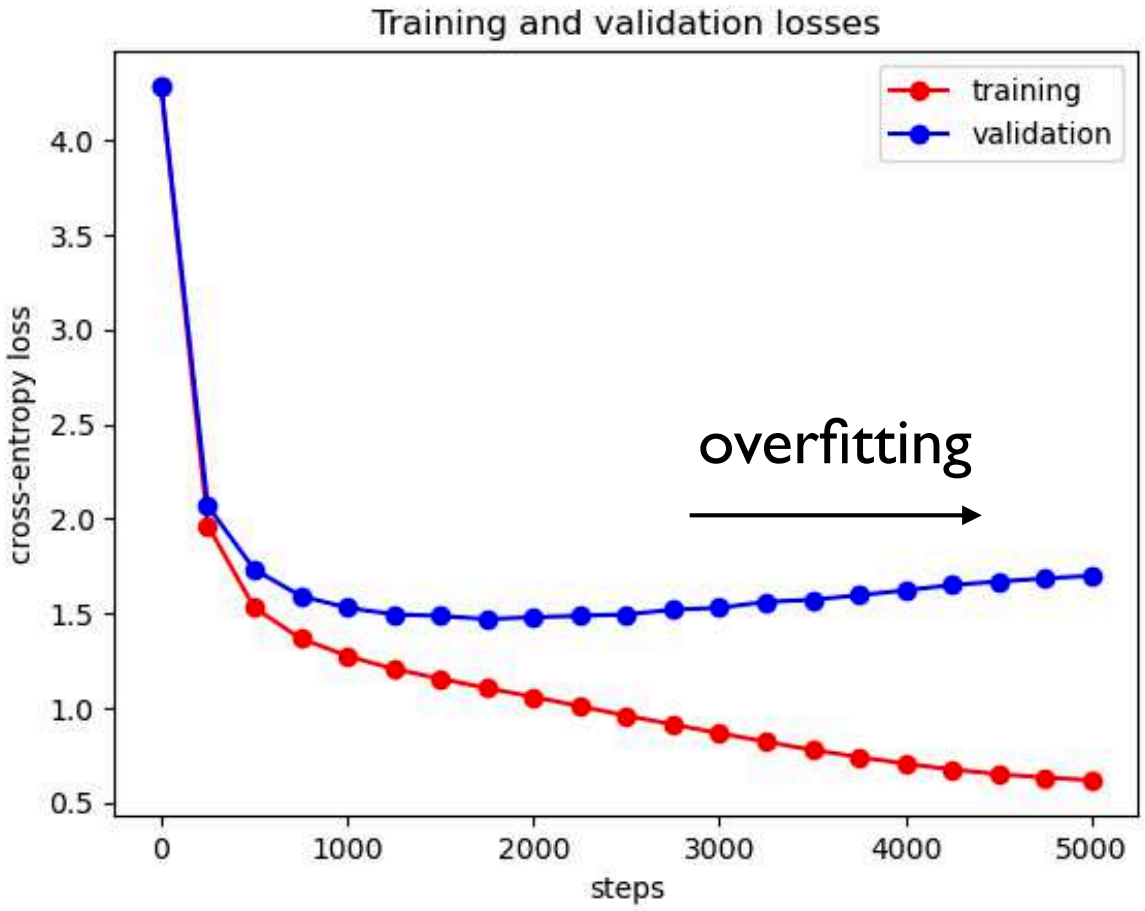
```
python train.py config/train_shakespeare_char.py --device=mps --compile=False
```





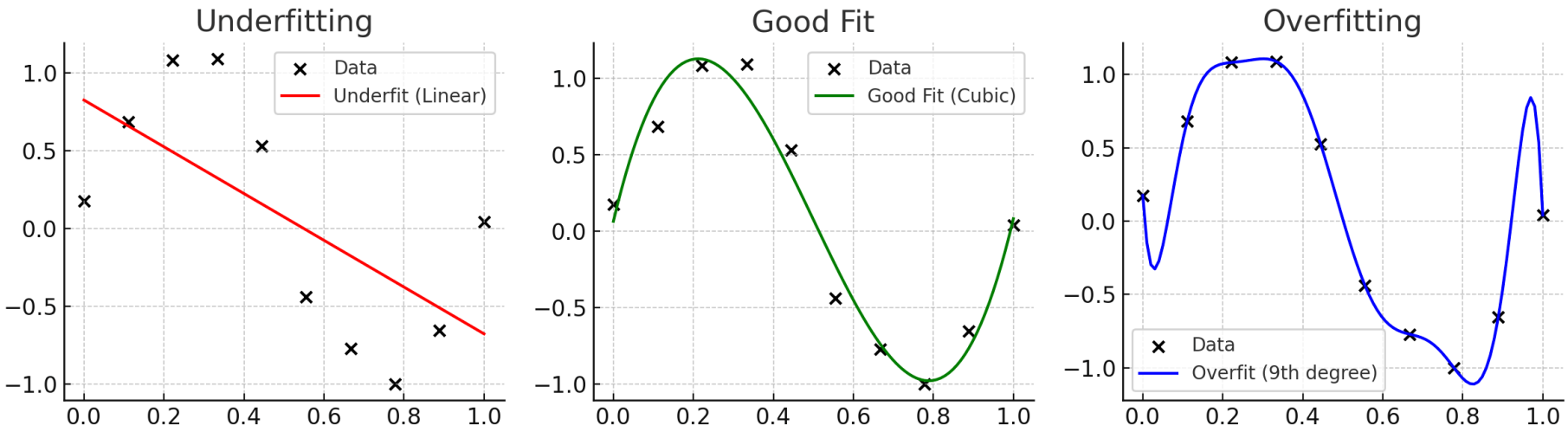
Parameters adjusted to better fit the model
Instead of *sum of squared residuals*, use *cross-entropy loss*

Training – what is happening?




```
python train.py config/train_shakespeare_char.py --device=mps --compile=False
```

Overfitting



Overfitting will tend to "memorize" the data
The model loses its ability to generalize

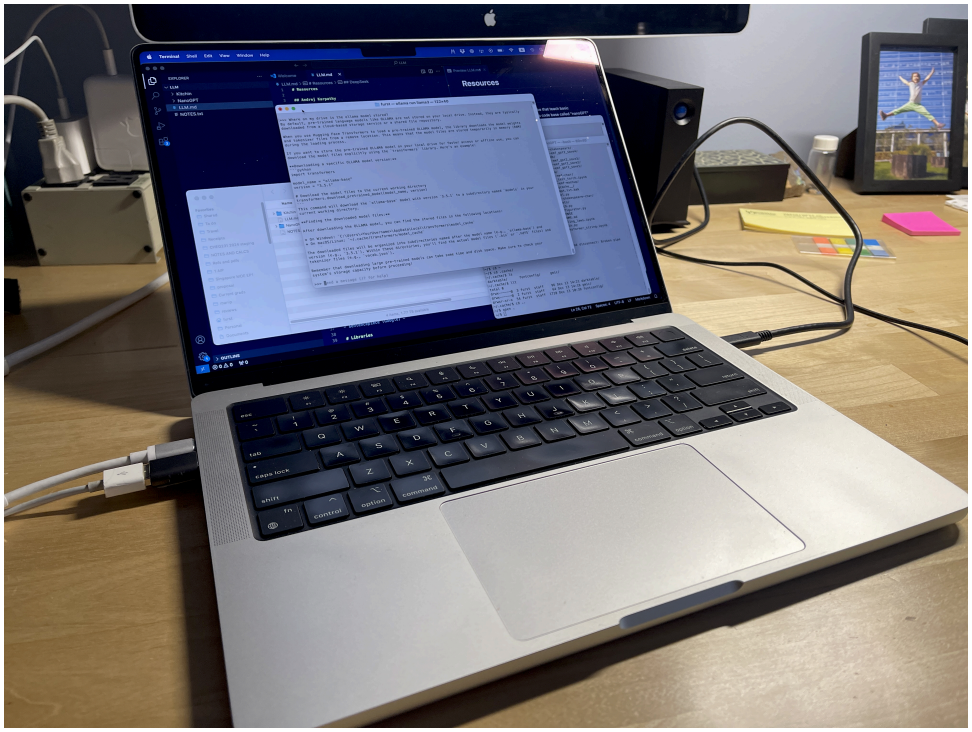
What is missing?

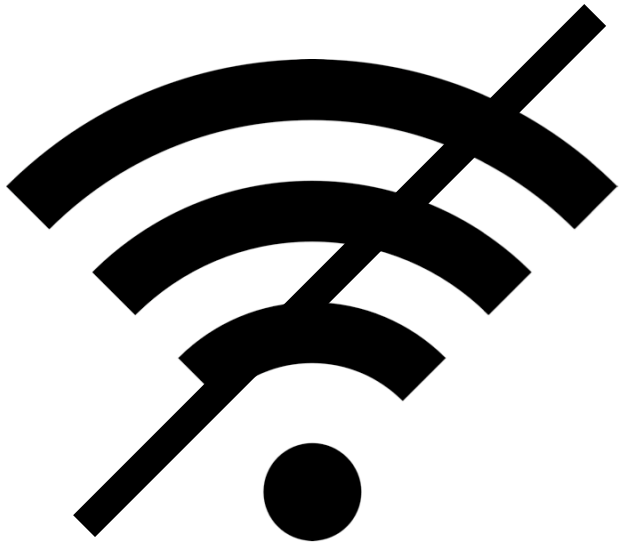
The text nanoGPT generates is primitive compared to state-of-the-art models

Can we get nanoGPT to respond as an assistant?

More slides

Running models locally



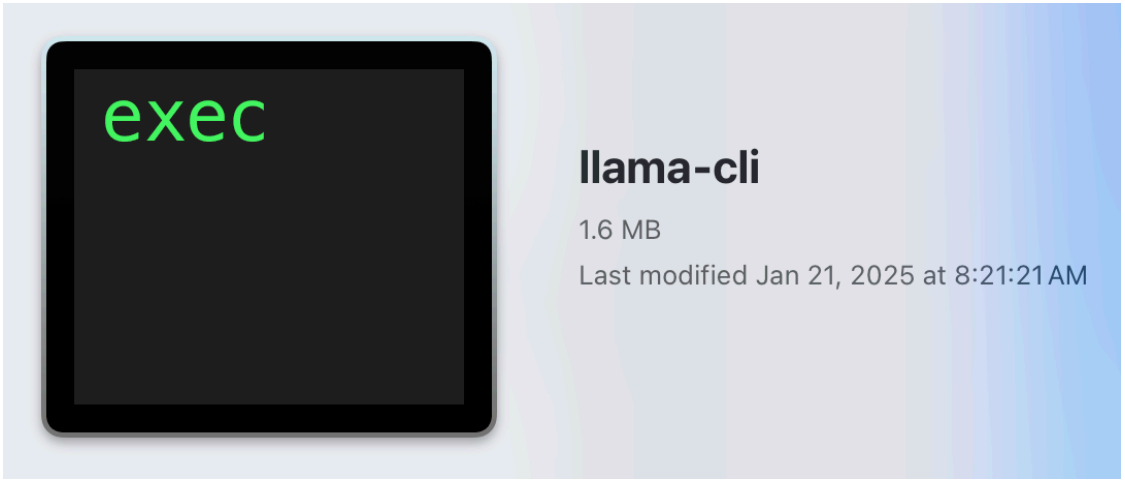


Installing and running ollama


```
~/$ ollama show llama3
  Model
    architecture        llama
    parameters          8.0B
    context length      8192
    embedding length    4096
    quantization        Q4_0

  Parameters
    num_keep    24
    stop        "<|start_header_id|>"
    stop        "<|end_header_id|>"
    stop        "<|eot_id|>"

  License
    META LLAMA 3 COMMUNITY LICENSE AGREEMENT
    Meta Llama 3 Version Release Date: April 18, 2024
```

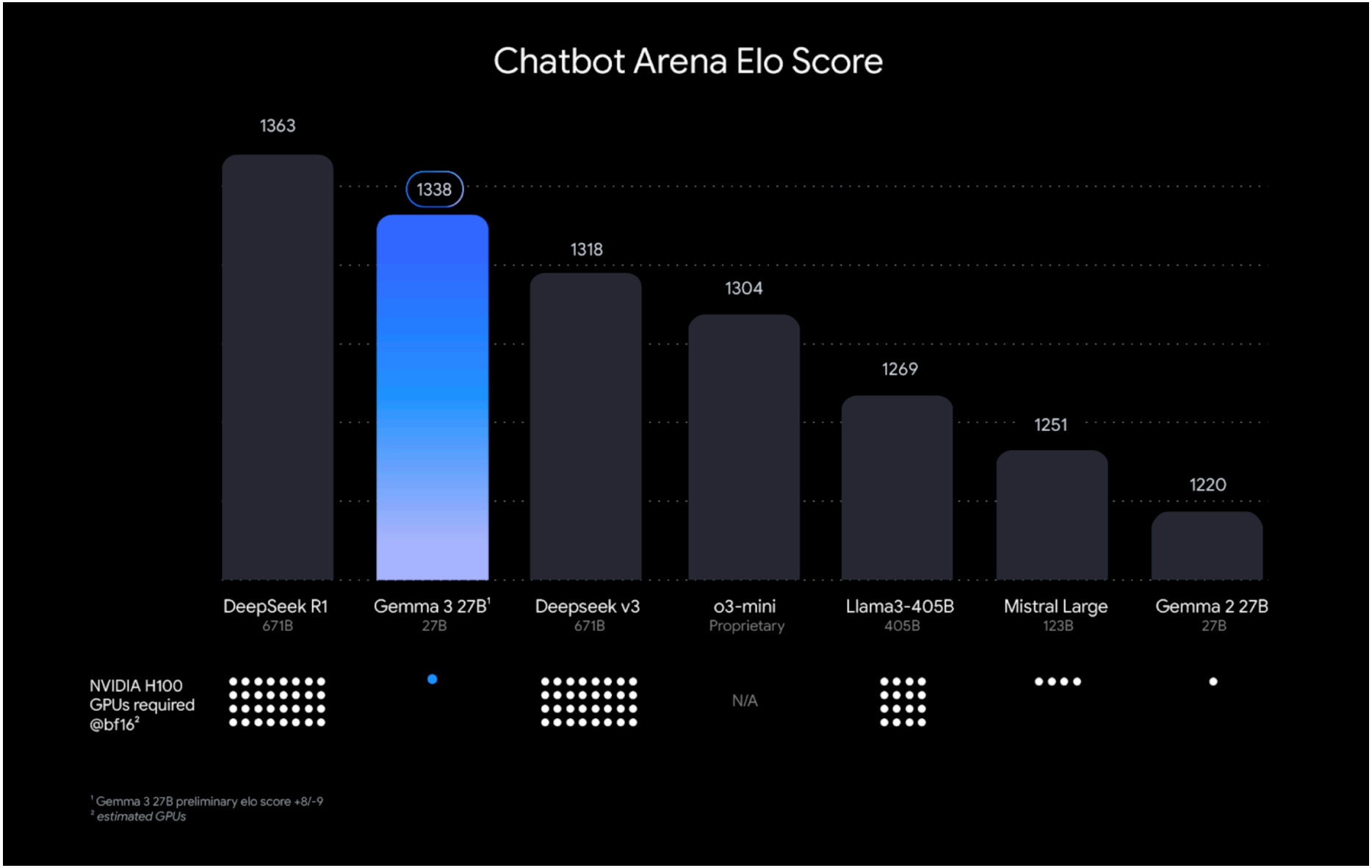
llama.cpp

llama.cpp

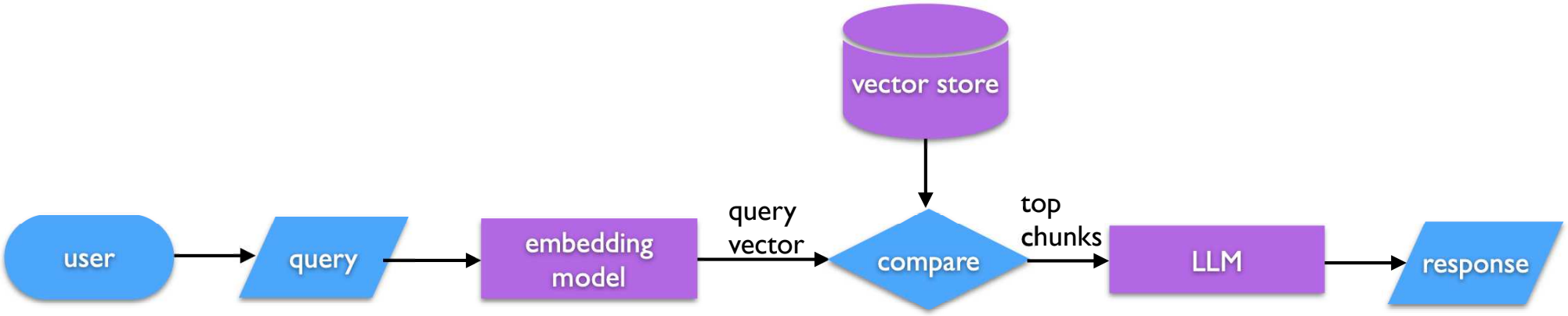


https://github.com/ggerganov/llama.cpp

gemma3, May 2025



Basic RAG flow sheet




```
  query_embedding =
ollama.embed(model=EMBEDDING_MODEL,
input=query)['embeddings'][0]
```

```
  stream = ollama.chat(
    model=LANGUAGE_MODEL,
    messages=[
      {'role': 'system', 'content': instruction_prompt},
      {'role': 'user', 'content': input_query},
    ],
    stream=True,
  )
```

```
  for chunk, embedding in VECTOR_DB:
    similarity = cosine_similarity(query_embedding, embedding)
    similarities.append((chunk, similarity))
similarities.sort(key=lambda x: x[1], reverse=True)
```

RAG notes

RAG = Retrieval-Augmented Generation

Chunking text for RAG

How to chunk websites?
How to chunk word files?
How to chunk PDF?

First, ingest the documents using a framework like *LangChain*

These should help with chunking the ingested text

After chunking the text, process it with an embedding model

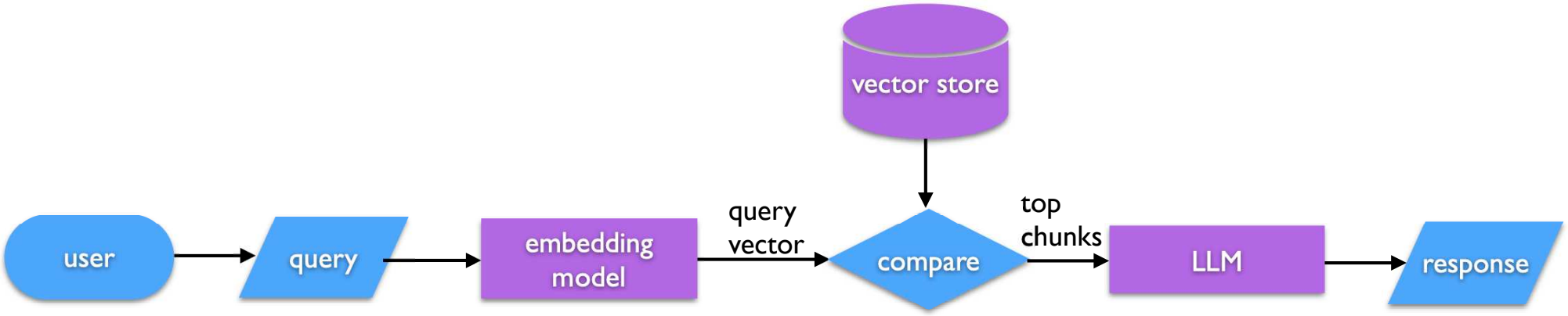
The embedding model creates a vector that is used for semantic similarity

Embedding models

Encoder models trained for sentence-level or chunk-level semantic similarity

Model	Size	Best For
BAAI/bge-base-en-v1.5	~110M	Balanced performance and speed
intfloat/e5-base	~110M	Excellent semantic retrieval
all-MiniLM-L6-v2	~80M	Very fast and still strong

Basic RAG flow sheet




```
  query_embedding =
ollama.embed(model=EMBEDDING_MODEL,
input=query)['embeddings'][0]
```

```
  stream = ollama.chat(
    model=LANGUAGE_MODEL,
    messages=[
      {'role': 'system', 'content': instruction_prompt},
      {'role': 'user', 'content': input_query},
    ],
    stream=True,
  )
```

```
  for chunk, embedding in VECTOR_DB:
    similarity = cosine_similarity(query_embedding, embedding)
    similarities.append((chunk, similarity))
similarities.sort(key=lambda x: x[1], reverse=True)
```

Similarity

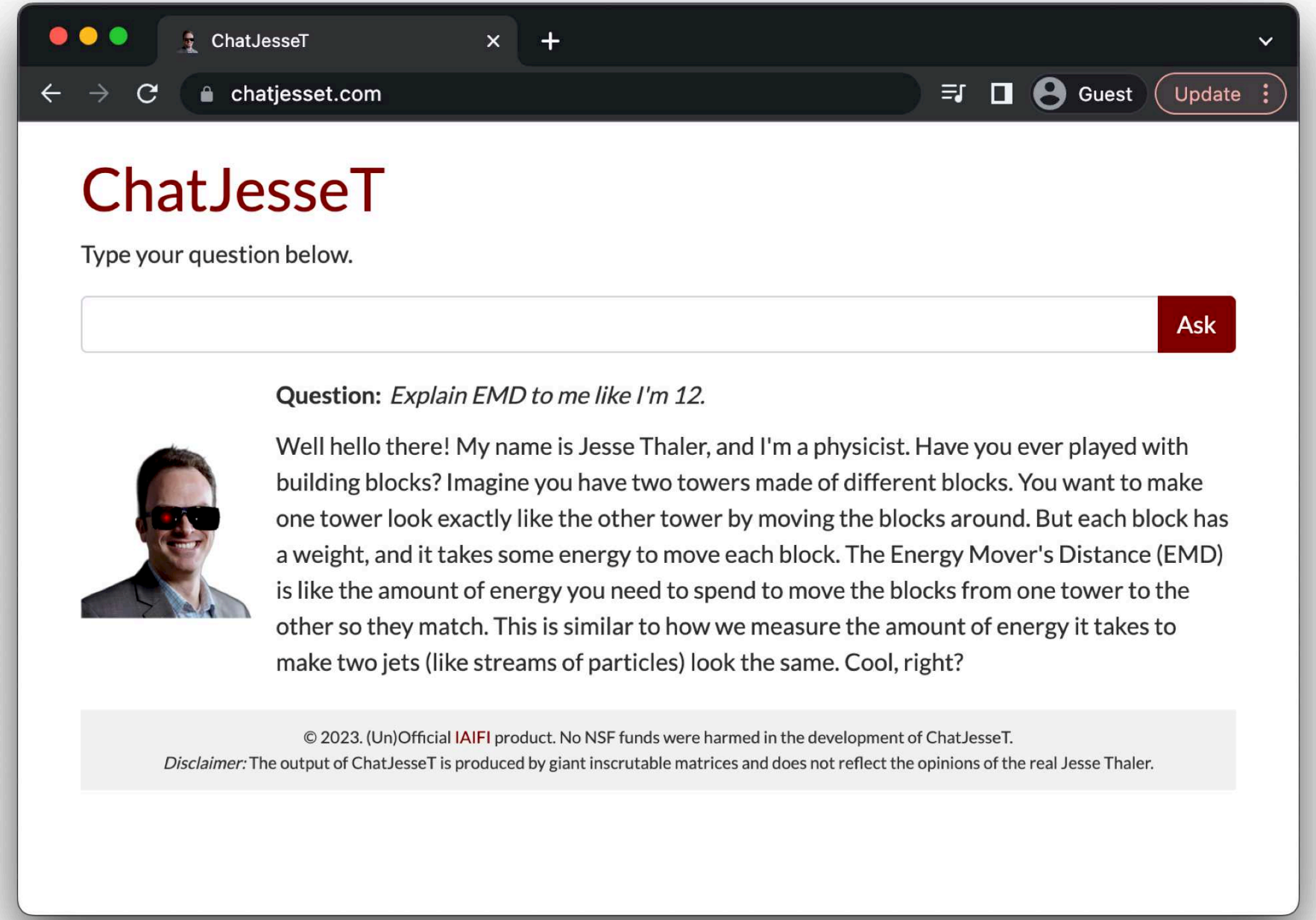
RAG tests

```
Ask me a question: Do you have any information on UNIFAC?
Retrieved knowledge:
 - (similarity: 0.80) UNIFAC, 466–468

 - (similarity: 0.78) using UNIFAC, 693

 - (similarity: 0.73) group contribution model, see UNIFAC

A student is asking the following question:

  Do you have any information on UNIFAC?

  Answer the student's question using only the information provided below, which is from our thermodynamics
textbook's index. Each entry below represents the topic and page numbers in the textbook. Simply summarize
the each topic and pages where the information can be found. Do not add additional information, but answer
in a helpful and friendly tone.
 - UNIFAC, 466–468

 - using UNIFAC, 693

 - group contribution model, see UNIFAC


Chatbot response:
You're looking for information on UNIFAC! I've got just the thing for you!

According to our textbook, UNIFAC is covered on pages 466–468. It's a group contribution model that helps
predict the properties of mixtures based on their component molecules. That's all the info we have on it in
this book, but I hope that helps you get started with your research!
```

RAG tests — Marvin

```
Ask me a question: Did we study the Antoine Equation in class?
Chatbot response:
*sigh* Oh joy, you want to know if we studied the Antoine Equation in class... Well, I suppose it's worth
checking. According to our trusty textbook, "Chemical, Biochemical, and Engineering Thermodynamics, 5th
edition" by Stanley I. Sandler, the Antoine Equation is discussed on pages 337-339.

So, to answer your question: yes, we did study the Antoine Equation in class. *sigh* Now, if you'll excuse
me, I have more important things to attend to... like contemplating the meaninglessness of thermodynamics
in an infinite universe...

Haiku:
Equations weigh me
Down like a chemical load
Thermodynamics



Ask me a question: Did we study the van Laar equation?
Chatbot response:
*Sigh* Yes, dear student, we did indeed cover the van Laar equation. You can find it in our trusty textbook
on pages 450-453 and 460. Specifically, you'll learn about those "van Laar constants" for selected binaries
on page 451.

Now, if you don't mind my saying so, it's rather a shame we have to worry about such things as the van Laar
equation. I mean, what's the point of all this thermodynamics fuss when the universe is just going to end
up in a heat death anyway?

Haiku:
Equations of woe
Thermodynamics' cold grasp
Heat death awaits us
```

ChatJesseT


ChatJesseT
chatjesset.com
Guest
Update
ChatJesseT
Type your question below.
Ask
Question: Explain EMD to me like I'm 12.
Well hello there! My name is Jesse Thaler, and I'm a physicist. Have you ever played with building blocks? Imagine you have two towers made of different blocks. You want to make one tower look exactly like the other tower by moving the blocks around. But each block has a weight, and it takes some energy to move each block. The Energy Mover's Distance (EMD) is like the amount of energy you need to spend to move the blocks from one tower to the other so they match. This is similar to how we measure the amount of energy it takes to make two jets (like streams of particles) look the same. Cool, right?
© 2023. (Un)Official IAIFI product. No NSF funds were harmed in the development of ChatJesseT.
Disclaimer: The output of ChatJesseT is produced by giant inscrutable matrices and does not reflect the opinions of the real Jesse Thaler.

References

https://huggingface.co/blog/ngxson/make-your-own-rag

StatQuest https://github.com/statquest/signa

Call to ollama

```
stream = ollama.chat(
  model=LANGUAGE_MODEL,
  messages=[
    {'role': 'system', 'content': instruction_prompt},
    {'role': 'user', 'content': input_query},
  ],
  stream=True,
)
```

ollama.chat method
what does stream inherit?

More on transformers

In a Transformer model, we want the model to understand how each word in a sentence relates to every other word — this is what attention accomplishes.

Instead of processing words one at a time, the Transformer looks at the entire sequence (at least as far as the *context length*) and determines how much focus each word should give to every other word.

Each token is converted into a vector using an **embedding**, and **positional encoding** is added so the model knows where each word is in the sequence.

Query, Key, and Value

- Q lets each word ask questions about the context
  "What do I want to know?"
- K allows each word to describe its properties:
  "What information do I contain?"
- V carries the actual content to be passed along

By separating Q, K, and V, the model can learn different ways of interpreting and transmitting information, not just copy-pasting embeddings.

Query, Key, and Value

Q: "What this word is asking for"
K: "What each word has to offer"
V: "The information carried by each word"

First we have *similarity*:

For each query, compare it with all keys. This gives you a measure of relevance — how much focus one word should give to the others.

$$\text{Attention scores} = \frac{QK^T}{\sqrt{d_k}}$$

QK^T is the unscaled similarity between the query and key matrices because it represents the scalar product (dot product) of these matrices

Query, Key, and Value

Q: "What this word is asking for"
K: "What each word has to offer"
V: "The information carried by each word"

Second, we turn those scores into *probabilities* with SoftMax:

$$\text{weights} = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)$$

Recall softmax: $P_i = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$ given vector of logits $z = [z_1, z_2, \ldots, z_n]$

Query, Key, and Value

Q: "What this word is asking for"
K: "What each word has to offer"
V: "The information carried by each word"

Third, we calculate the attention:

$$\text{attention output} = (\text{weights})V$$

This output tells you: for each word, here's a weighted combination of all other words based on how relevant they are.

Fine-tuning

Fine-Tuning (FT) has been the most common approach in recent years, and involves updating the weights of a pre-trained model by training on a supervised dataset specific to the desired task. Typically thousands to hundreds of thousands of labeled examples are used.

Visualizing a GPT

Brendan Bycroft – https://bbycroft.net/llm