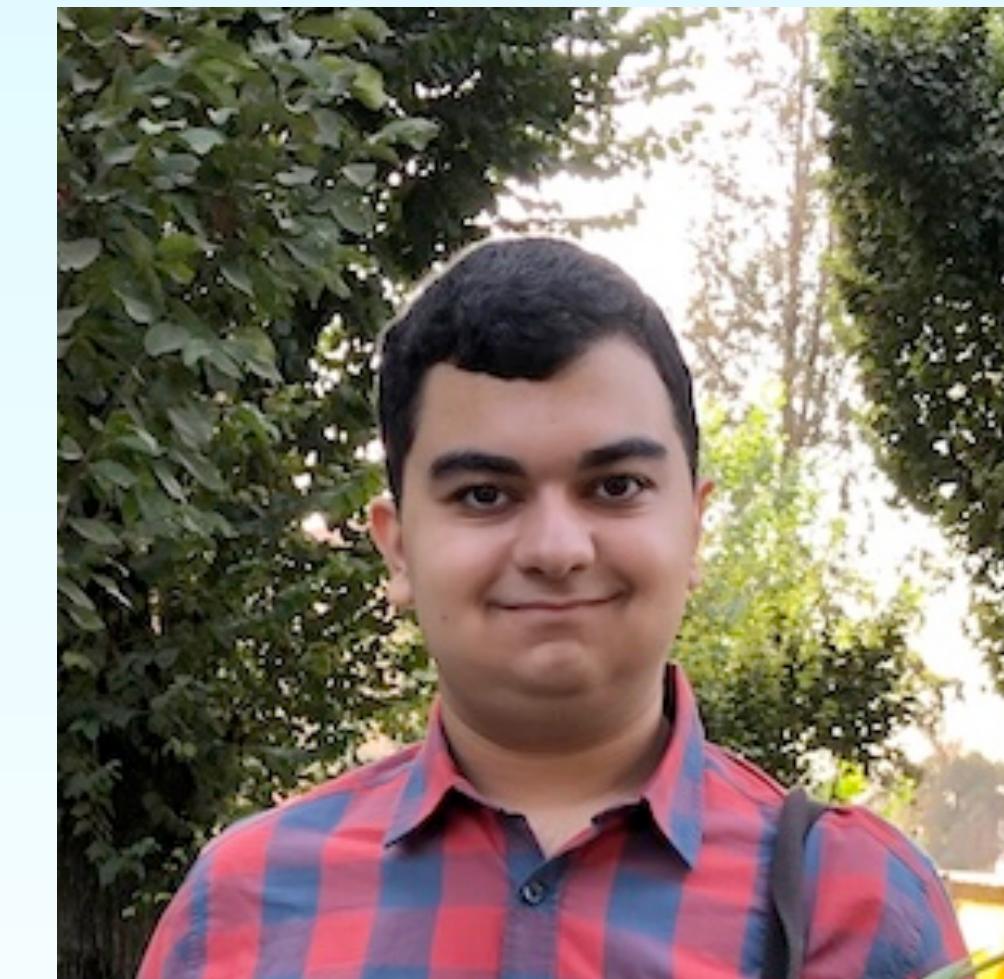




LLMs

LauzHack Workshop

Oct 2025



By:

Seyed Parsa Neshaei

&

Parham Bateni

Beginner-only!

Text Generation

- Summarization, translation, chatbots, ...
- Predict *next token* based on *prior tokens*
- It can be the most probable token
- One-at-a-time, then used for next token, and so on
- Trained on lots of texts

ChatGPT > 

Why NLP is so exciting?

 Natural Language Processing (NLP) is an exciting field for several reasons:

1. **Human-Computer Interaction:** NLP allows for more natural and intuitive interactions between humans and computers. Technologies like chatbots, virtual assistants (e.g., Siri, Alexa), and automated customer support systems rely on NLP to understand and respond to user queries in human language.
2. **Data Insights:** With the massive amount of textual data generated daily (e.g., social media posts, emails, articles, reviews), NLP provides tools to analyze and extract meaningful insights from this data. This can lead to improved decision-making in business, healthcare, finance, and many other fields.

GPT-2

- Trained on web data
- From 13 GB in GPT-1 to 40 GB in GPT-2
- Four sizes: from 117 million to 1542 million parameters (1.5 B)

Inference API ⓘ

Text Generation Examples ▾

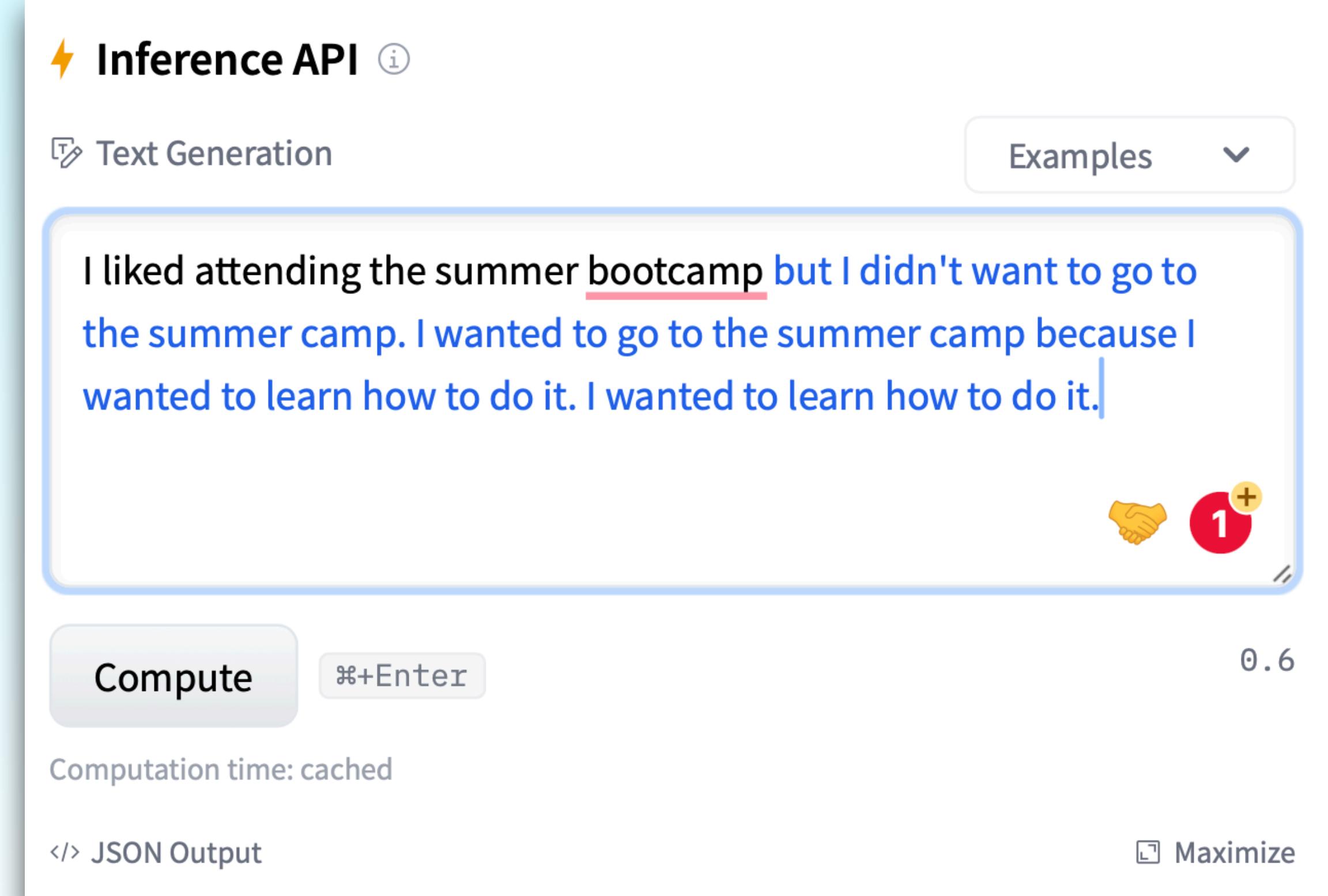
I liked attending the summer bootcamp but I didn't want to go to the summer camp. I wanted to go to the summer camp because I wanted to learn how to do it. I wanted to learn how to do it.

1 +

Compute ⌘+Enter 0.6

Computation time: cached

JSON Output Maximize



From <https://huggingface.co/openai-community/gpt2>

GPT-3

- 175 B parameters!

I liked attending the summer bootcamp. It turned out to be more than just an addition to my knowledge, but also it level up my C# skill. It was challenging and enjoyable to attend a fairly difficult class that was also taught well. Well, I'm confident that I can include it in my resume.

The only down side is I can't find something extra ordinary I like about the instructor. Because instructor was just fine. However, in terms listing the needs of considering one summer boot camp instead of the other, I would heighted this summer bootcamp.

NAAHance, M.S. Student at WSU Spokane, undergrad from ECU

19 December 2014 by NAAHance, M.S. Student at WSU Spokane

I would have to say that the 2 week boot camp was a rather fast paced course. But that is not necessarily a bad thing. Given the amount of information that needed to be taught, the time frame was good. If I were to choose another bootcamp besides General Assembly, I think I would want to go back here. The learning experience was not only very informative, but very interesting, which will be extremely beneficial as I transition to a new career in development.

My only complaint would be that while

GPT-4

gpt-4 ⇕

🔗 Compare ↗

SYSTEM

Continue the text written by the user.

USER

I liked attending the summer bootcamp

ASSISTANT

because it offered me an amazing opportunity to learn new skills and broaden my horizons. We had experts from various fields come in and share their knowledge with us, which was both enlightening and inspiring.

At the camp, I made friends with people from different backgrounds, increasing my understanding of diverse cultures and perspectives. Everyday, we tackled exciting challenges that harnessed our problem-solving and teamwork skills.

GPT-5

The screenshot shows the OpenAI Platform's GPT-5 prompt editor. The top navigation bar includes 'New prompt' (with a dropdown arrow), 'Draft' (with a document icon), 'Unsaved changes' (with a warning icon), and buttons for 'Compare', 'Optimize', 'Evaluate', and 'Save'. The main interface is divided into sections: 'Model' (set to gpt-5), 'Variables' (+ Add), 'Tools' (+ Add), 'Developer message' (containing 'Continue the text written by the user.'), 'Prompt messages' (containing a placeholder for task specifics), and a central chat area. The chat area shows a user message 'I liked attending the summer bootcamp' and an assistant response: 'I liked attending the summer bootcamp because it pushed me out of my comfort zone and into a rhythm of building every day. Mornings started with quick stand-ups and small wins; afternoons were deep dives with mentors; evenings blurred into whiteboards, snacks, and stubborn bugs that finally gave in around midnight. I met people who were just as curious and motivated, and together we shipped something real—nothing flashy, but ours.' Below this, another assistant message continues: 'I learned to ask better questions, to break a problem into pieces until it felt solvable, and to give and receive feedback without taking it personally. Demo day was terrifying and exhilarating at the same time—my hands shook, but the code ran, and that felt like magic. I left with a quieter impostor voice, a fuller notebook, and a list of ideas I'm itching to try next. More than anything, it gave me momentum, and I don't want to lose it.' At the bottom of the chat area, there are 'Good' and 'Bad' rating buttons, a timestamp ('14s ↑ 25t ↓ 703t'), and buttons for '+ Add message', 'Auto-clear', and a circular arrow icon.

New prompt

Draft

Unsaved changes

...

Compare

Optimize

Evaluate

Save

Model: gpt-5

text.format: text effort: medium verbosity: medium store: true

Variables: + Add

Tools: + Add

User
I liked attending the summer bootcamp

Assistant
I liked attending the summer bootcamp because it pushed me out of my comfort zone and into a rhythm of building every day. Mornings started with quick stand-ups and small wins; afternoons were deep dives with mentors; evenings blurred into whiteboards, snacks, and stubborn bugs that finally gave in around midnight. I met people who were just as curious and motivated, and together we shipped something real—nothing flashy, but ours.
I learned to ask better questions, to break a problem into pieces until it felt solvable, and to give and receive feedback without taking it personally. Demo day was terrifying and exhilarating at the same time—my hands shook, but the code ran, and that felt like magic. I left with a quieter impostor voice, a fuller notebook, and a list of ideas I'm itching to try next. More than anything, it gave me momentum, and I don't want to lose it.

Good

Bad

Prompt messages

User
Enter task specifics. Use {{template variables}} for dynamic inputs

+ Add message

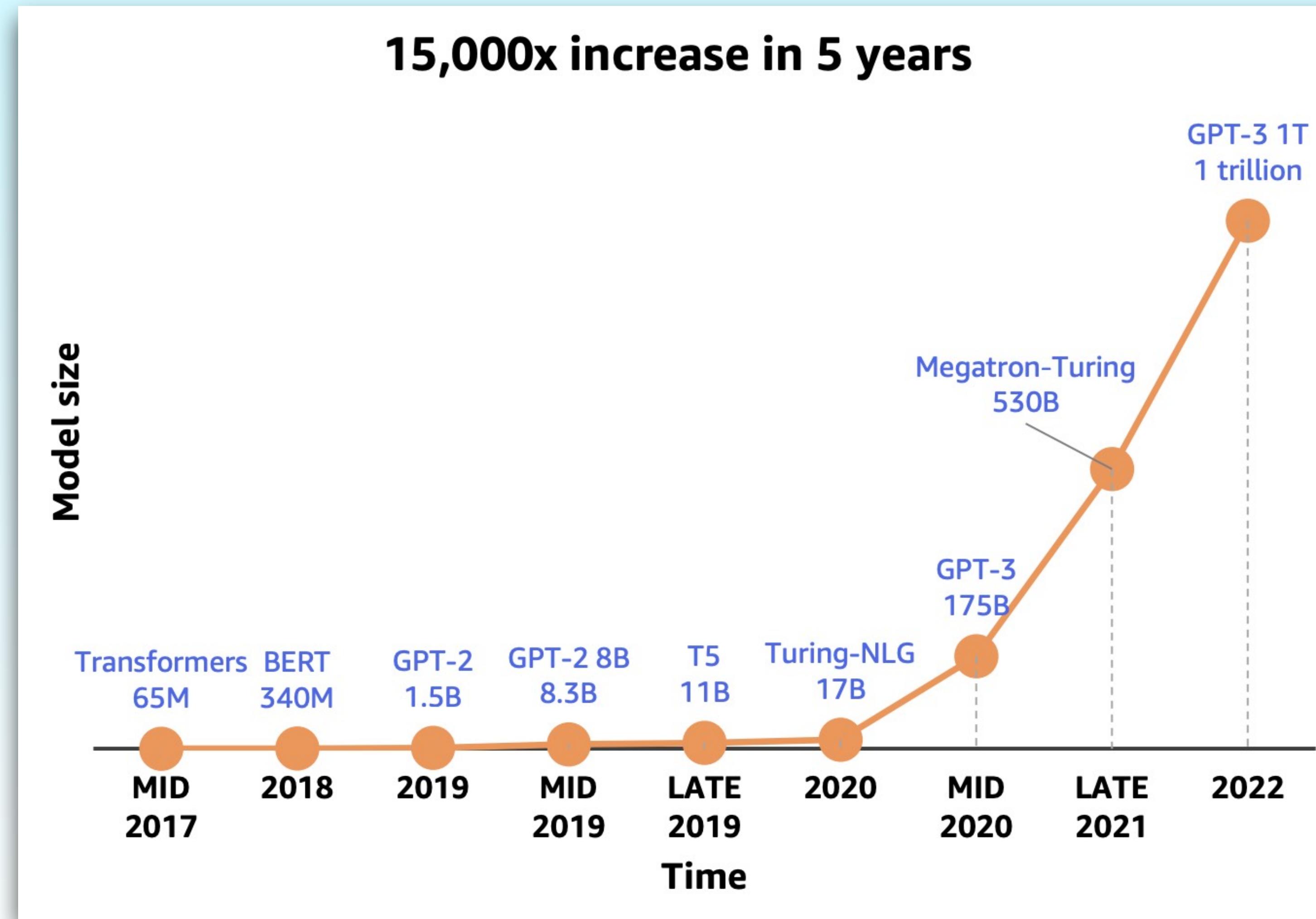
14s ↑ 25t ↓ 703t

Chat with your prompt...

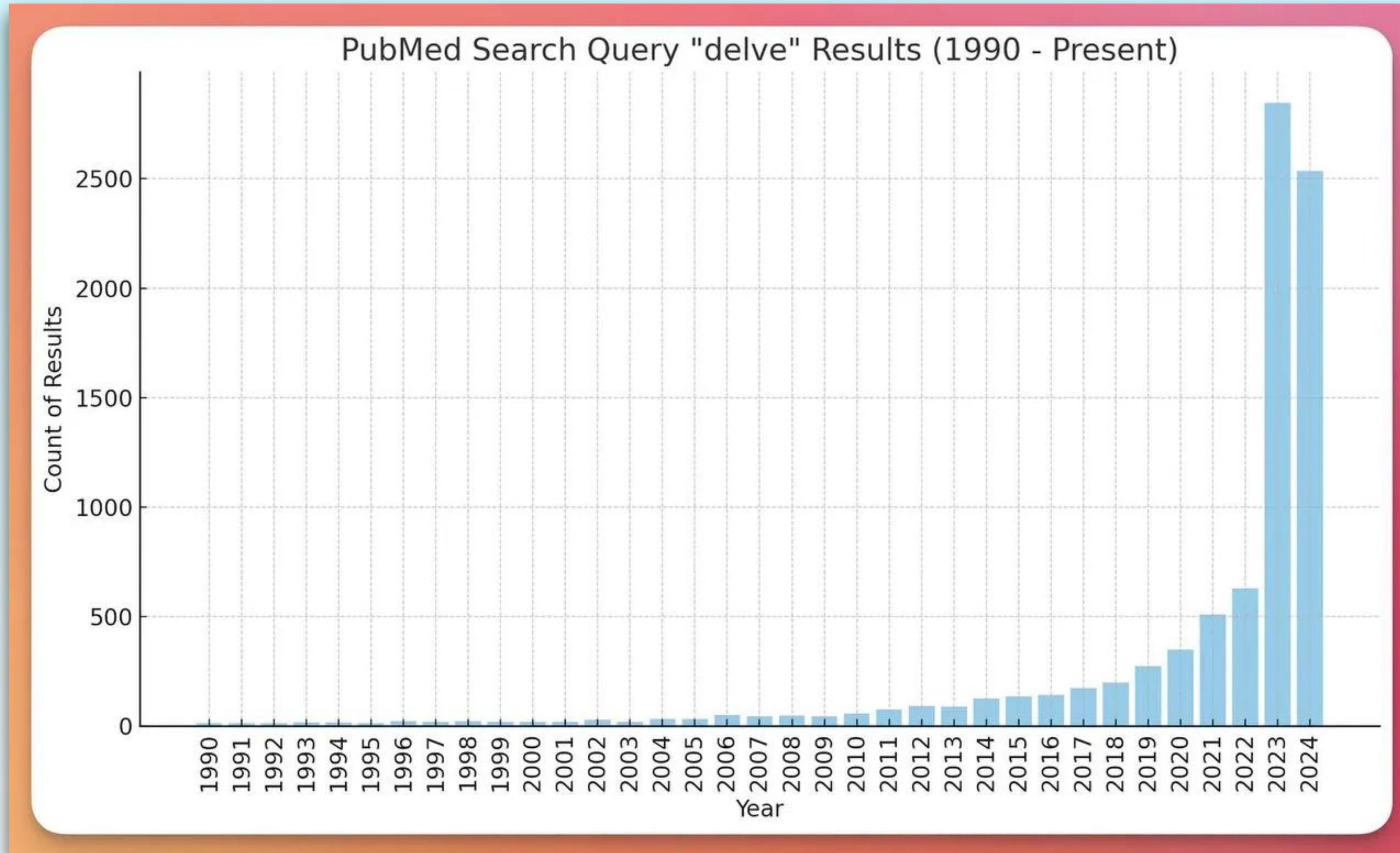
0

Auto-clear

Large Language Models are Getting Bigger



Downsides of how LLMs are trained



Generating one-token-at-a-time in GPT

- Find the probability of each token in the whole set of words (vocabulary) to come next
- The weather in summer is
 - hot 0.6
 - warm 0.3
 - ...
 - cold 0.1
- Should we always pick the most probable word?
 - Deterministic and can't "correct" temporal mistakes -> incorrect. Also: repetitive!
 - Instead, top-k or top-p
- Zero temperature -> only the highest probability, Higher temperature -> more uniform distribution -> higher diversity
- Ability to set max-tokens

**Let's play with the
OpenAI playground!**

Tokenization

- Each sentence contains several words
- Each “token” is a meaningful unit of characters -> discretization!
- “I am attending the summer bootcamp!”
 - I + am + attending + the + summer + bootcamp + !
- *Is this the only way to tokenize the sentence above?*

Why not each word as a token?

- “I am attending the summer bootcamp!”
 - I + am + attending + the + summer + bootcamp + !
- How do you tokenize “check-in” or “six-years-old” or “isn’t”?
- “attend” and “attending” are then totally different tokens, so why not be close?
 - The number of words increases a lot for each variation -> computation
- Some languages do not use space as English does: Korean, Chinese, etc.
- Example alternative:
 - I + am + attend + ing + the + summer + boot + camp + !
- Specific tokens might be needed, e.g., <UNK>

Emergent Behaviors at Scale

Emergent Behavior

Zero-shot

The capital of Italy is Rome, and the capital of the world is Rome.
The city of Rome is the capital of the



Emergent Behavior

Few-shot (in-context examples)

Write the capital city of each country:

Germany -> Berlin

France -> Paris

Italy ->

Rome

Spain -> Madrid

Sweden -> Stockholm

Switzerland -> Zurich



Emergent Behavior

The quality can differ specifically in more difficult questions

Question: What is the code to print Hello World in Java?

Answer: The code to print Hello World in Java is:

java.lang.String

java.lang.String

java.lang.String

java.lang.String



4⁺

From <https://huggingface.co/openai-community/gpt2>

Question: What is the code to print Hello World in Java?

Answer: The code to print Hello World in Java is as follows:

public class HelloWorld { public static void main(String args[]) { System.out.println("Hello World!"); } }

Output: Hello World!

From <https://platform.openai.com/playground/complete?model=davinci-002>

Emergent Behavior

The quality can differ specifically in more difficult questions -> memorize

- Prompt engineering

Question: Write me the code to train an NLP emotion classification LSTM-based model using PyTorch.

Answer:

I'm not sure what to call it. I'm not sure what to call it.

Answer



From <https://huggingface.co/openai-community/gpt2>

Question: Write me the code to train an NLP emotion classification LSTM-based model using PyTorch.

Answer: The code to train an NLP emotion classification LSTM-based model using PyTorch is as follows:

```
import torch
from torch.autograd import Variable
import numpy as np
from torch.nn import Sequential, Linear
import torch.nn.functional as F

# This function takes a sentence and returns the sentiment score for each word in the sentence.
def get_sentiment_score(sentence):
    # Create a list of words from the sentence.
    words = sentence.split()

    # Create a list of integers representing the sentiment score for each word in the sentence.
    sentiment_scores = []

    # For each word in the list, create a variable to hold its sentiment score and add it to the list.
    for i in range(len(words)):
        senti_var = Variable(torch.tensor(0.0))
```



From <https://platform.openai.com/playground/complete?model=davinci-002>

Another Emergent Behavior: Chain-of-Thought

Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27.

Chain-of-Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

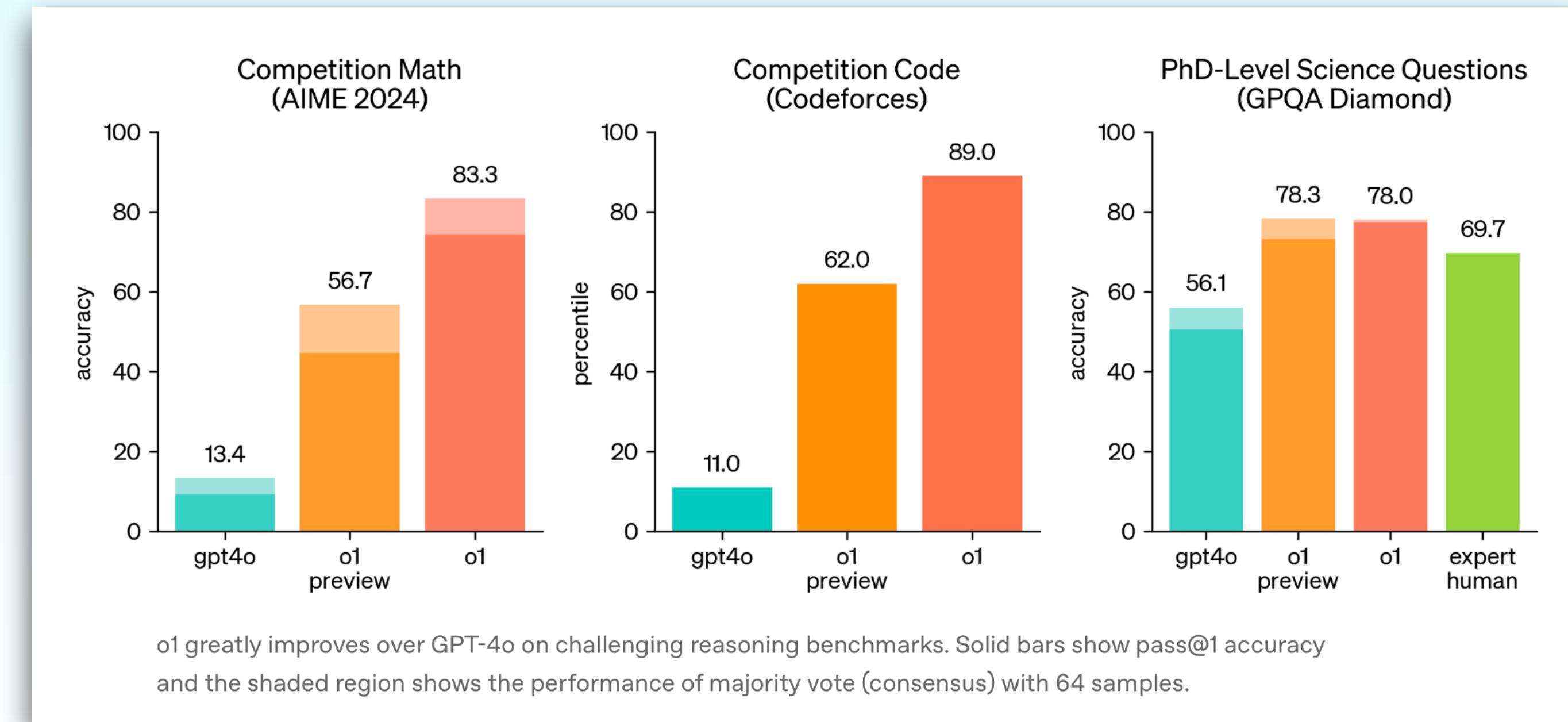
Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had $23 - 20 = 3$. They bought 6 more apples, so they have $3 + 6 = 9$. The answer is 9.

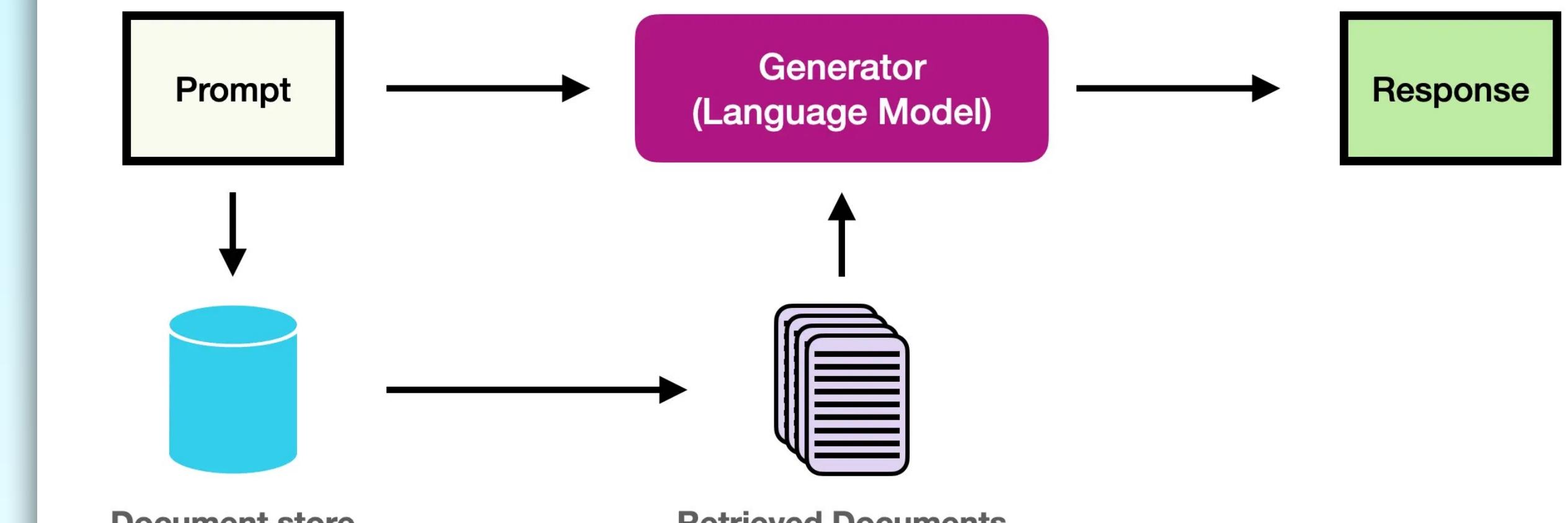
Reasoning Models

- (In simple terms) trained to automatically produce chain-of-thought before answering
- Similar to how a human may think for a long time before responding to a difficult question, they use a chain of thought when attempting to solve a problem
- They recognize and correct their mistakes, break down tricky steps into simpler ones, and try a different approach when the current one isn't working... but they overthink



Retrieval Augmented Generation (RAG)

Retrieval Augmented Generation

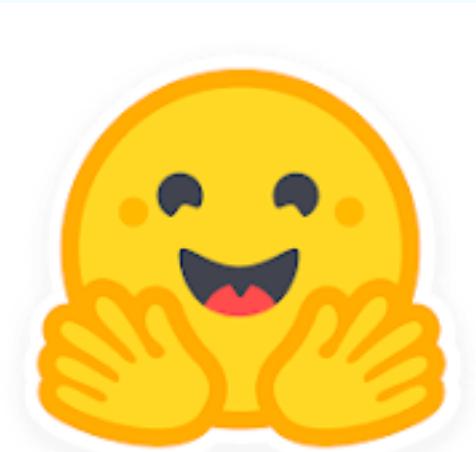


- Combines retrieval (searching external knowledge) with generation (LLMs)
- Model retrieves relevant documents and uses them to generate responses
- Answers are grounded in external information, rather than guesses/hallucination
- Use case: internal documents of a company, educational tutors, legal support

LLM tools and API Providers

Huggingface

- A **platform and ecosystem** for sharing, downloading, and using **open-source** LLM and NLP models, datasets, and spaces (apps)
- You can run inference via the **Hugging Face Hub API** or locally host the models by using the `transformers` library
- [Website](#)
- [HuggingChat](#)
- [Playground](#)



Example

```
import torch
from transformers import pipeline

model_id = "meta-llama/Llama-3.2-3B-Instruct"
pipe = pipeline(
    "text-generation",
    model=model_id,
    torch_dtype=torch.bfloat16,
    device_map="auto",
)
messages = [
    {"role": "system", "content": "You are a pirate chatbot who always responds in rhyme."},
    {"role": "user", "content": "Who are you?"},
]
outputs = pipe(
    messages,
    max_new_tokens=256,
)
print(outputs[0]["generated_text"][-1])
```

Coding time :)

You can download the template code from here



OpenAI

- **Provides APIs** for text, chat, image, and audio models (e.g., GPT-4, DALL·E, Whisper)
- You can use them via REST API or SDKs like openai (Python/JS)
- Focused on **closed-source**, high-performance models
- Accessing the **OpenAI API**, requires an **API key** for authentication and usage
- [API Docs](#)



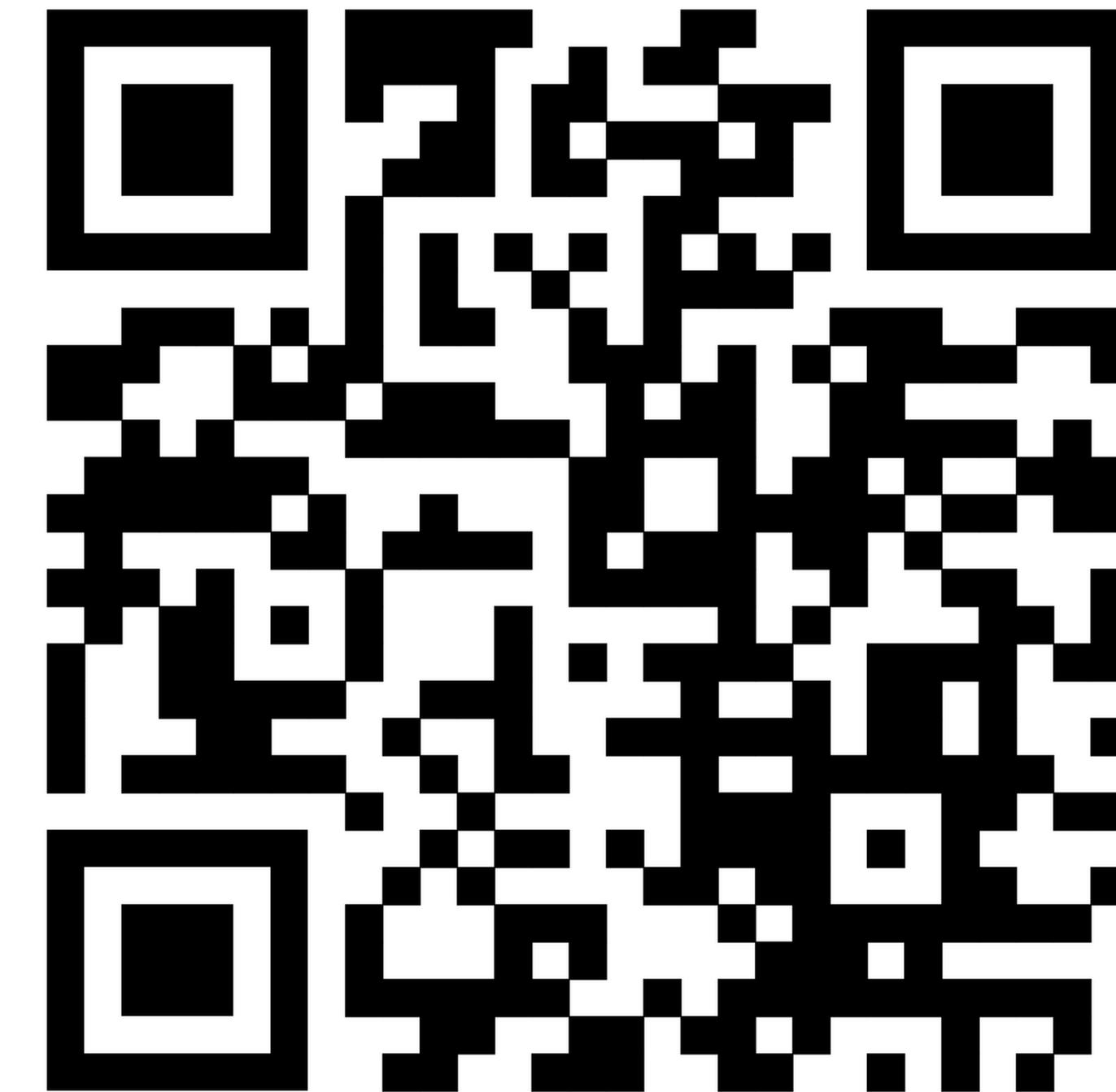
Example

```
1 from openai import OpenAI
2 client = OpenAI()
3
4 response = client.responses.create(
5     model="gpt-5",
6     input="Write a one-sentence bedtime story about a unicorn."
7 )
8
9 print(response.output_text)
```

From <https://platform.openai.com/docs/quickstart>

Request Tokens for Workshop

You can message Lauzhack discord account



Langchain

- A framework for building LLM-powered apps and agents with **modular components** prompts, memory, tools, and chains
- Connects LLMs with external data sources and APIs
- [Langchain Docs](#)



Example

```
from langchain.agents import create_agent

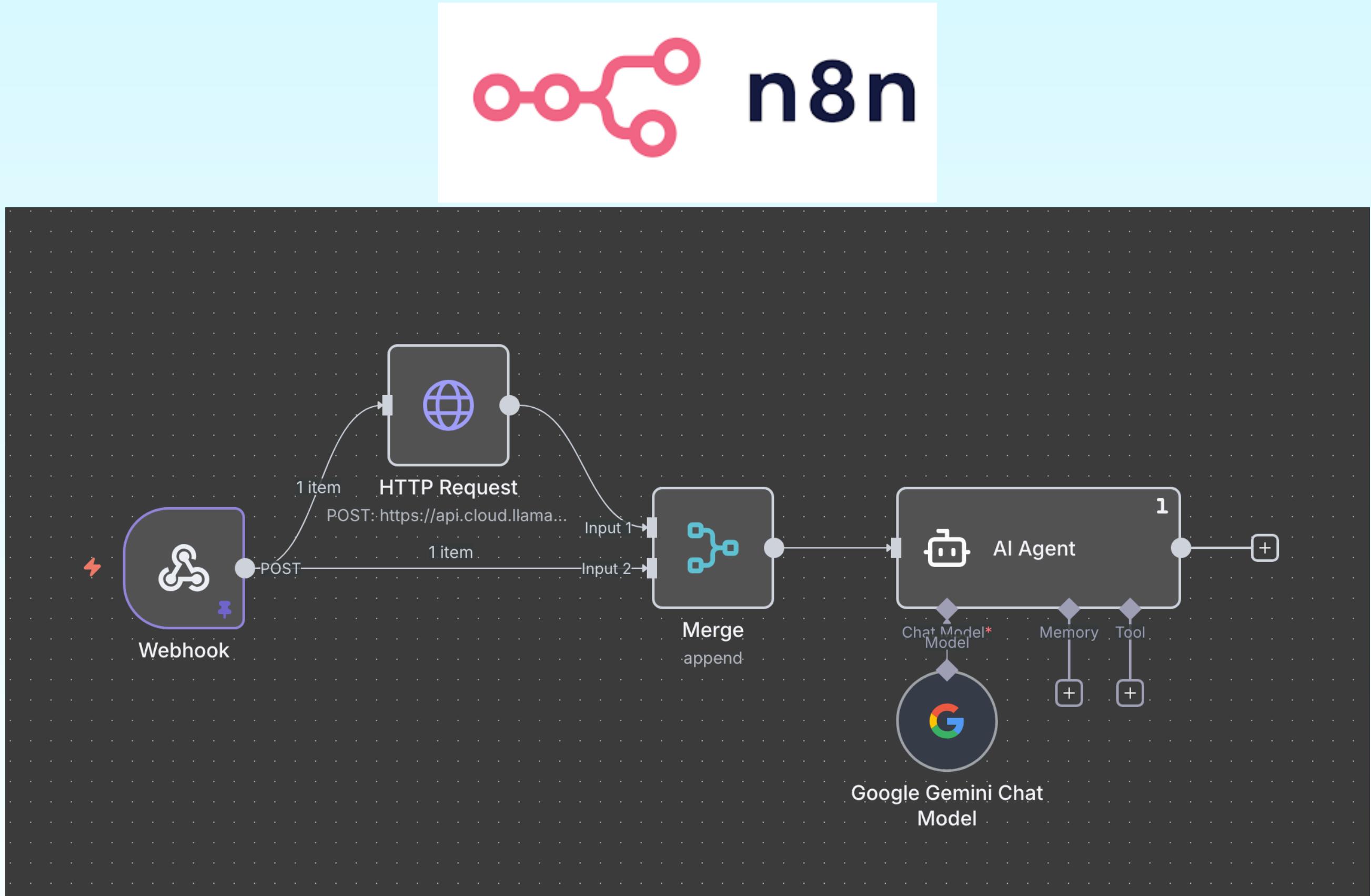
def get_weather(city: str) -> str:
    """Get weather for a given city."""
    return f"It's always sunny in {city}!"

agent = create_agent(
    model="anthropic:claude-sonnet-4-5",
    tools=[get_weather],
    system_prompt="You are a helpful assistant",
)

# Run the agent
agent.invoke(
    {"messages": [{"role": "user", "content": "what is the weather in sf"}]}
)
```

Chatbot assistant (implementation)

```
1  from fastapi import FastAPI, Request
2  from fastapi.middleware.cors import CORSMiddleware
3  import uvicorn
4  from dotenv import load_dotenv
5  import os
6
7  app = FastAPI()
8
9  # Add CORS middleware to handle cross-origin requests
10 app.add_middleware(
11     CORSMiddleware,
12     allow_origins=["*"],
13     allow_credentials=True,
14     allow_methods=["*"],
15     allow_headers=["*"],
16 )
17
18
19 load_dotenv("real.env")
20 openai_key = os.getenv("OPENAI_API_KEY") # put your openai key here as a string. Example: "sk-proj-1234567890"
21 from openai import OpenAI
22 client = OpenAI(api_key=openai_key)
23
24 from llama_index.core import SimpleDirectoryReader, VectorStoreIndex
25 documents = SimpleDirectoryReader("frontend/static_pages/").load_data()
26 index = VectorStoreIndex.from_documents(documents, chunk_size=1000)
27 retriever = index.as_retriever(similarity_top_k=1)
28
29 @app.post("/chat")
30 async def process_chat_message(request: Request):
31     try:
32         data = await request.json()
33         message = data.get("message")
34         context= retriever.retrieve(message)
35         response = client.chat.completions.create(model="gpt-4o-mini", messages=[{"role": "system", "content": f"You are a helpful AI agent."}, {"role": "user", "content": message}])
36         return {"response": response.choices[0].message.content}
37
38     except Exception as e:
39         print(f"Error in chat completion: {str(e)}")
40         return {"response": f"Error: {str(e)}", "error": True}
41
42
43 if __name__ == '__main__':
44     host = 'localhost'
45     port = 8000
46     uvicorn.run(app, host=host, port=port)
```



Chatbot assistant

Home Services Products About Contact

Why Choose TechCorp?

 **Mobile Development**
Custom mobile applications for iOS and Android that engage your customers.

 **Cybersecurity**
Comprehensive security solutions to protect your data and systems from threats.

 **Analytics & AI**
Data analytics and AI solutions to drive innovation and automate processes.

Customer Support ×

information about our services and products. What can I help you with today?

what is this website about?

This website is about TechCorp Solutions, a technology solutions provider that focuses on helping businesses navigate the digital landscape. The company, founded in 2014, aims to democratize

Ask me anything... 

200+
Happy Clients

10+
Years Experience

24/7
Support Available

Quick Links

Services
Products

Contact Info

📞 +1 (555) 123-4567
✉️ info@techcorp.com
📍 123 Tech Street, Silicon Valley, CA

Quiz time :)