LLMs

LauzHack Workshop

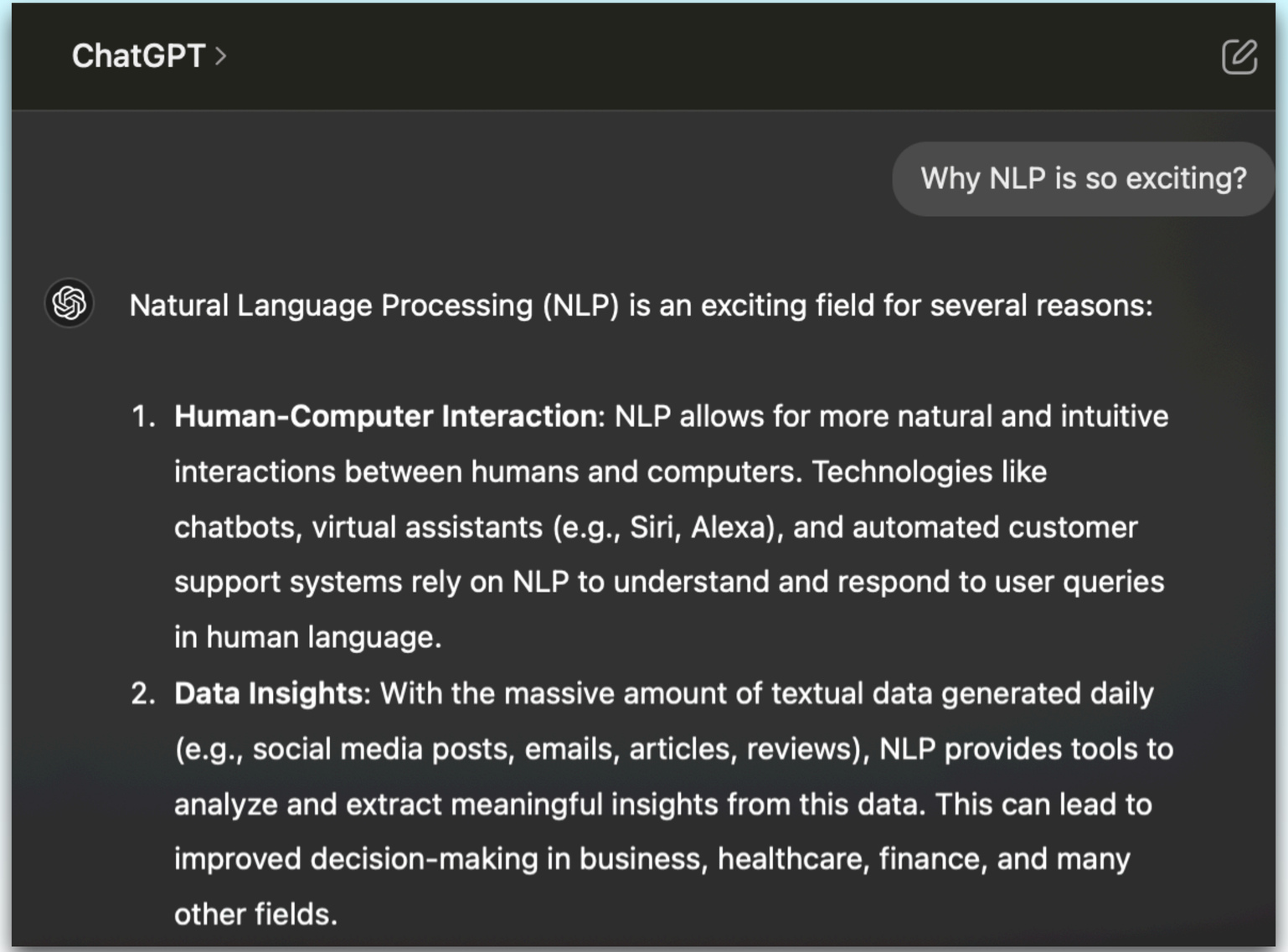
Oct 2025

Several examples and material are from or inspired by the CS-552 Modern NLP lecture slides offered at EPFL by Prof. Antoine Bosselut

Beginner-only!

Text Generation

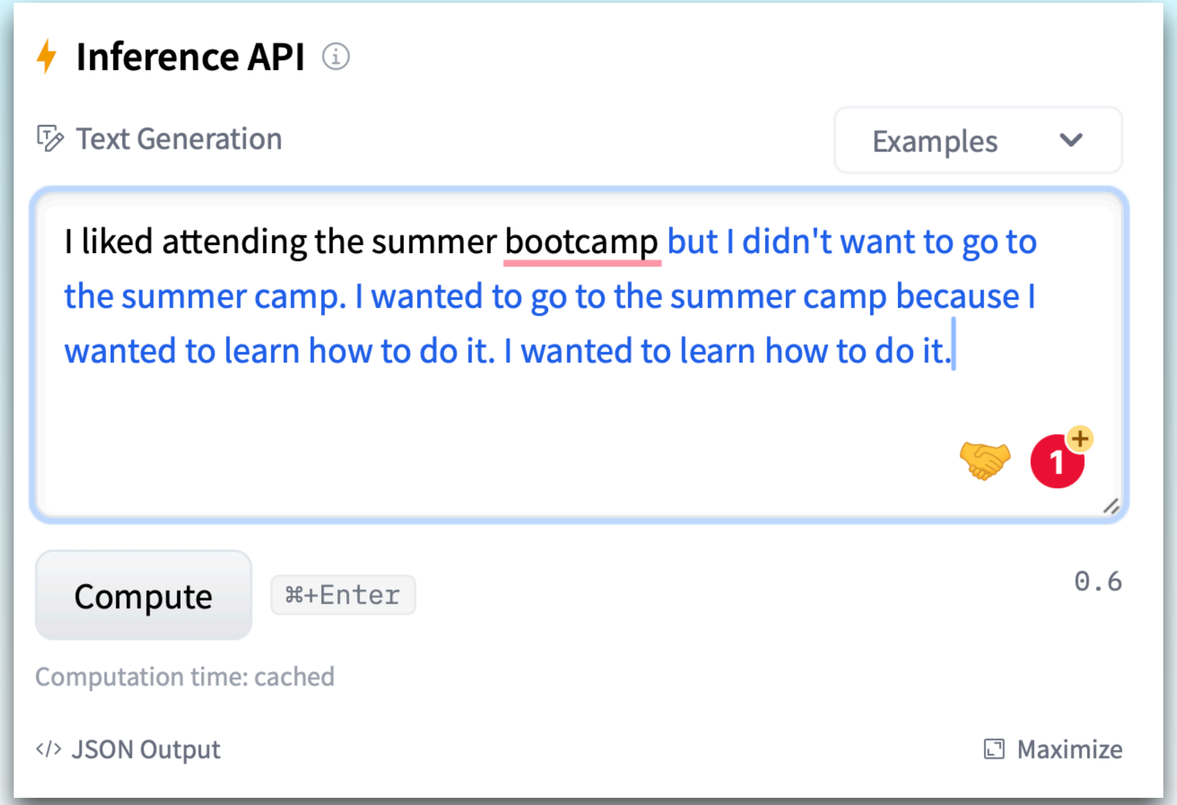
- Summarization, translation, chatbots, …
- Predict *next token* based on *prior* tokens
- It can be the most probable token
- One-at-a-time, then used for next token, and so on
- Trained on lots of texts



From the ChatGPT macOS app

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)



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.

N A A H a n c e, M.S. Student at WSU Spokane, undergrad from ECU

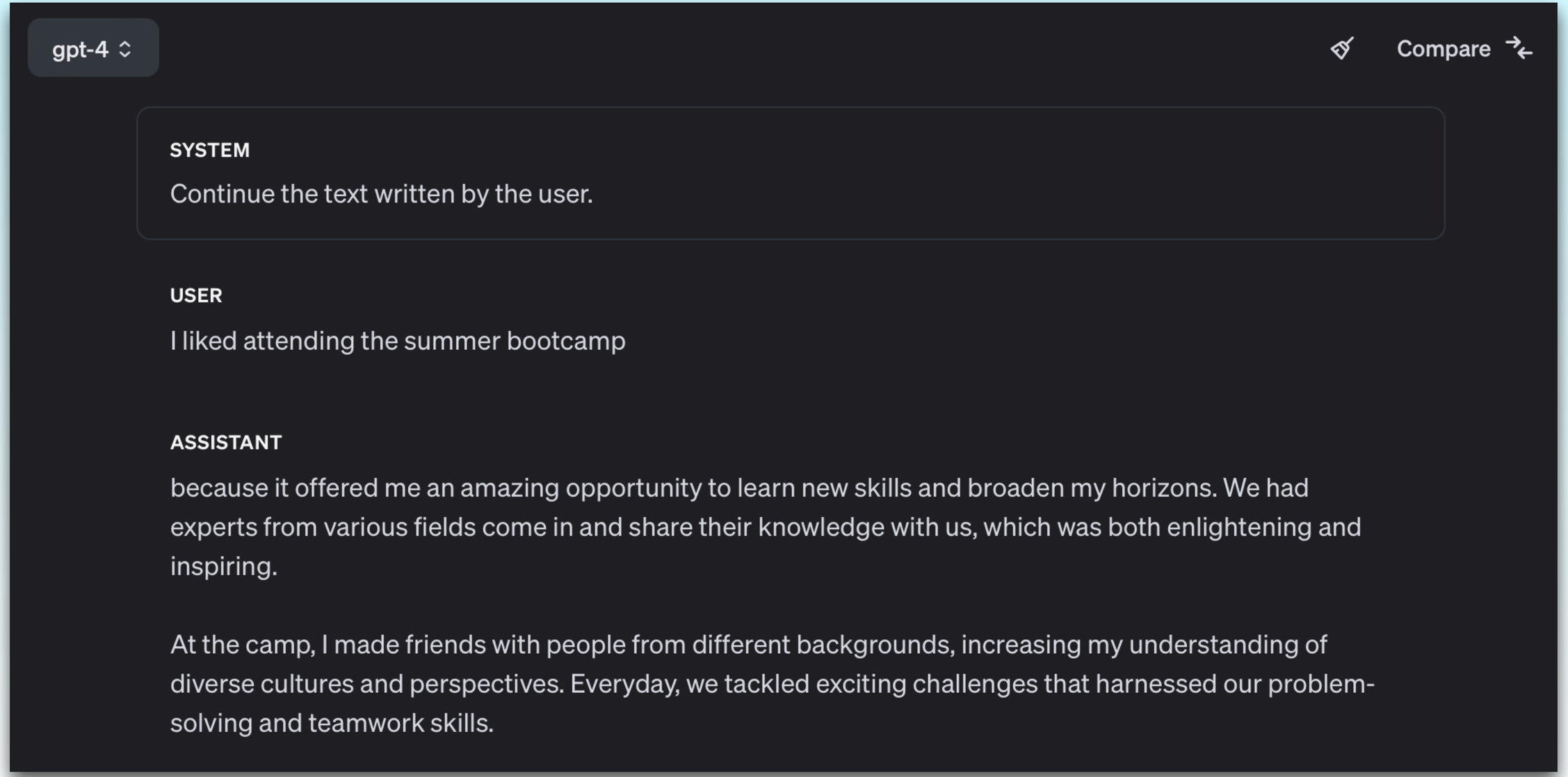
19 December 2014 by N A A H a n c e, M.S. Student at WSU Spokane

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

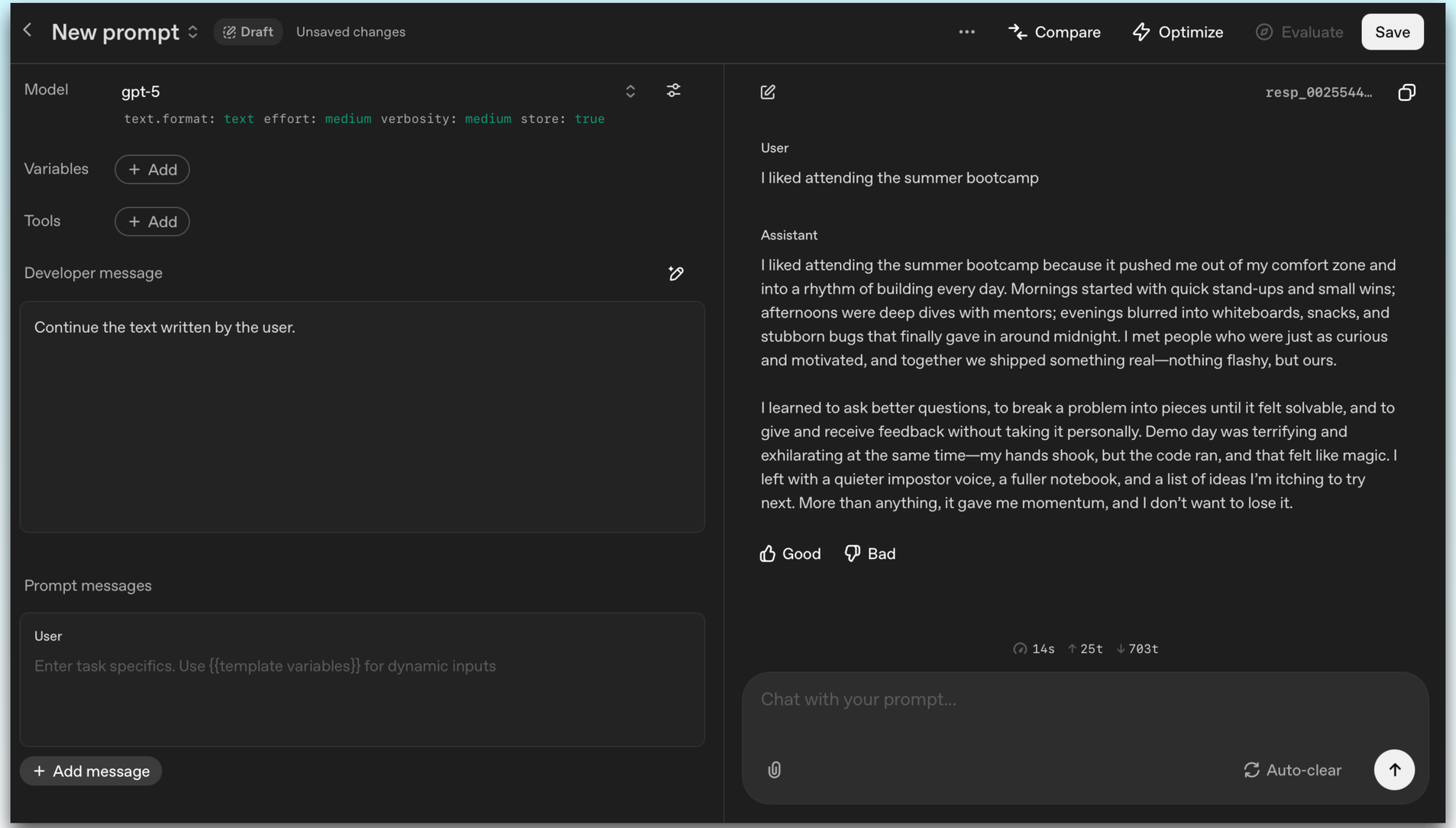
From https://platform.openai.com/playground/complete?model=davinci-002

GPT-4



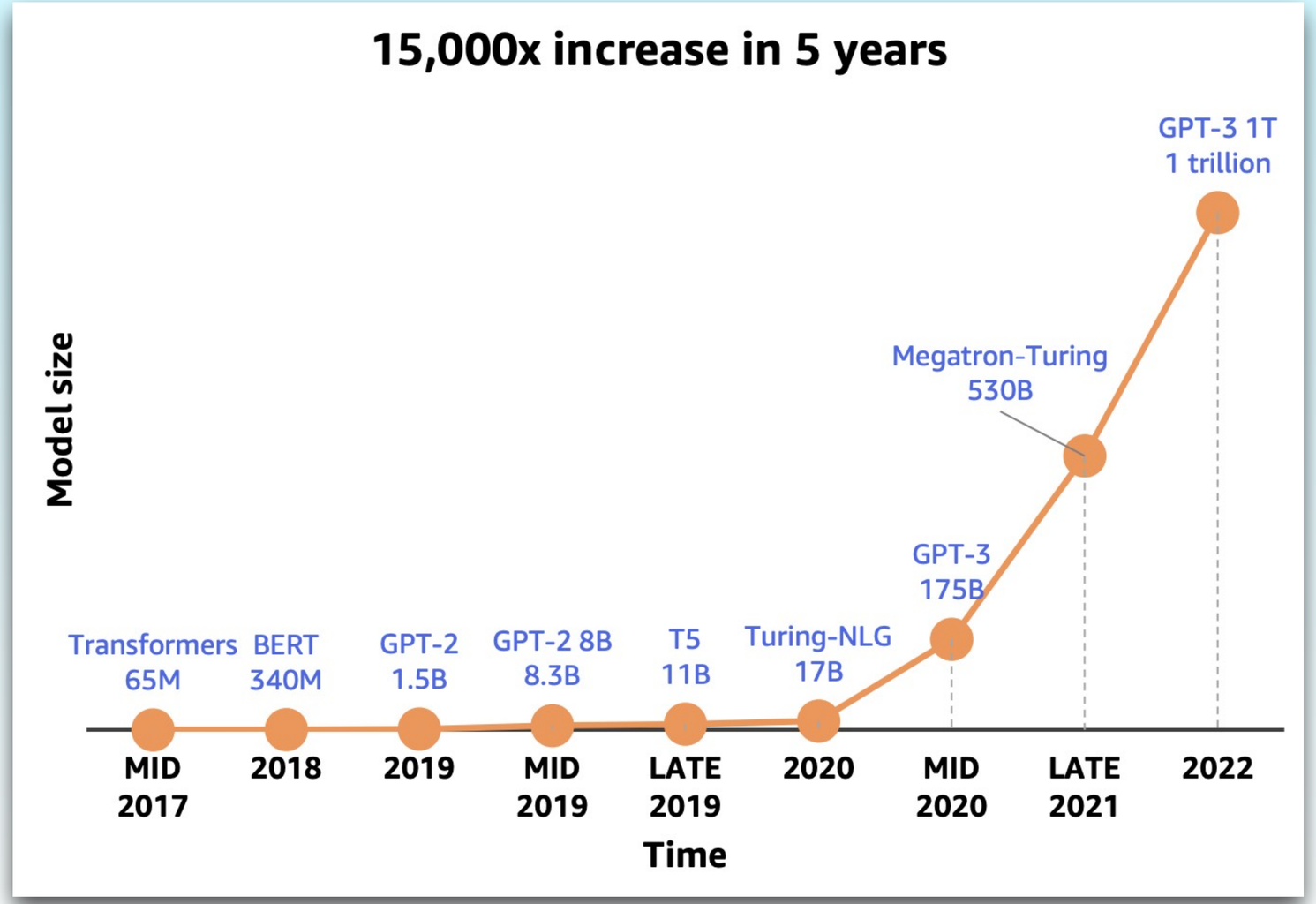
From https://platform.openai.com/playground/chat?models=gpt-4

GPT-5



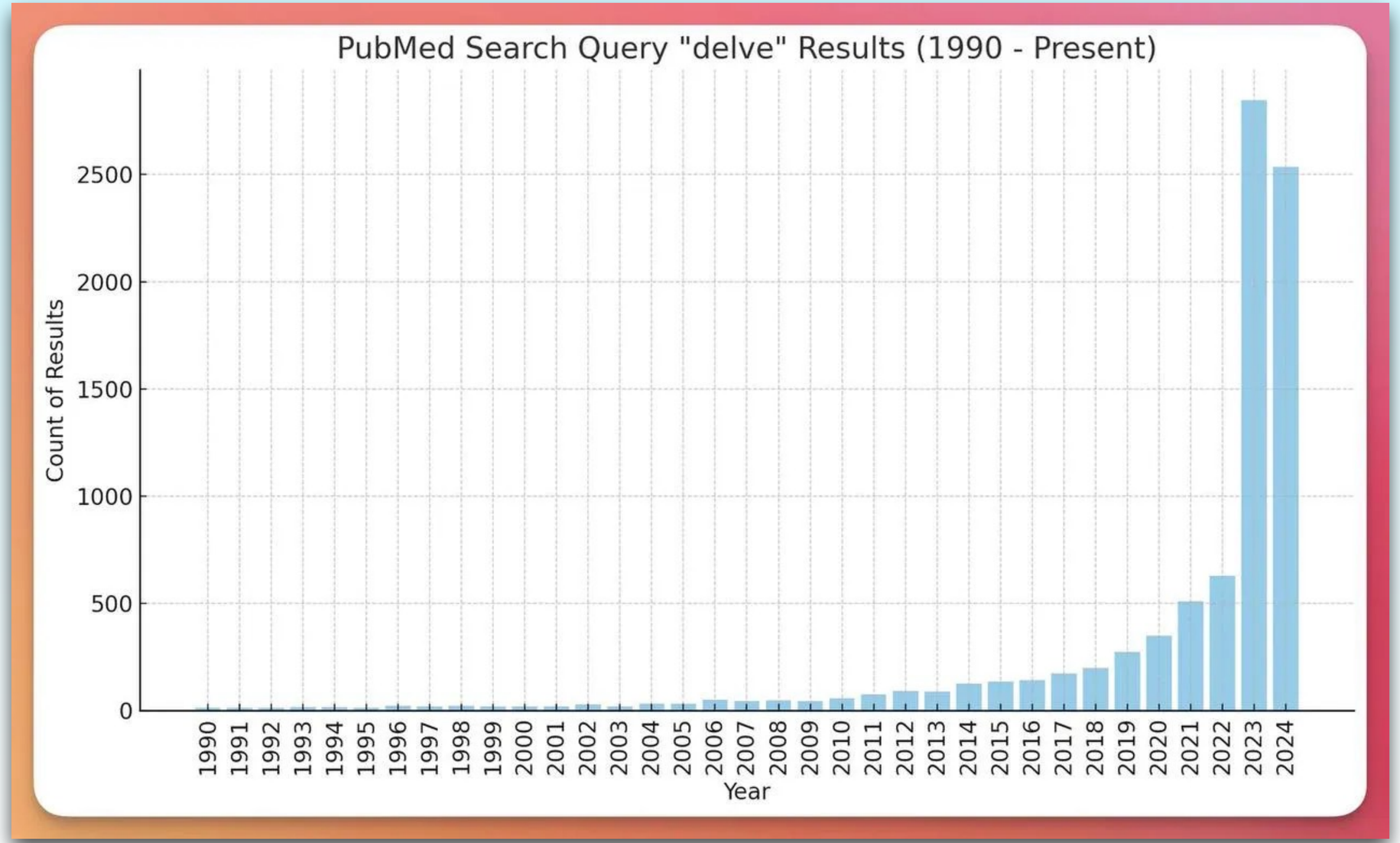
From https://platform.openai.com/chat/edit?models=gpt-5

Large Language Models are Getting Bigger



https://neptune.ai/blog/nlp-models-infrastructure-cost-optimization

Downsides of how LLMs are trained



https://hesamsheikh.substack.com/p/why-does-chatgpt-use-delve-so-much

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?*

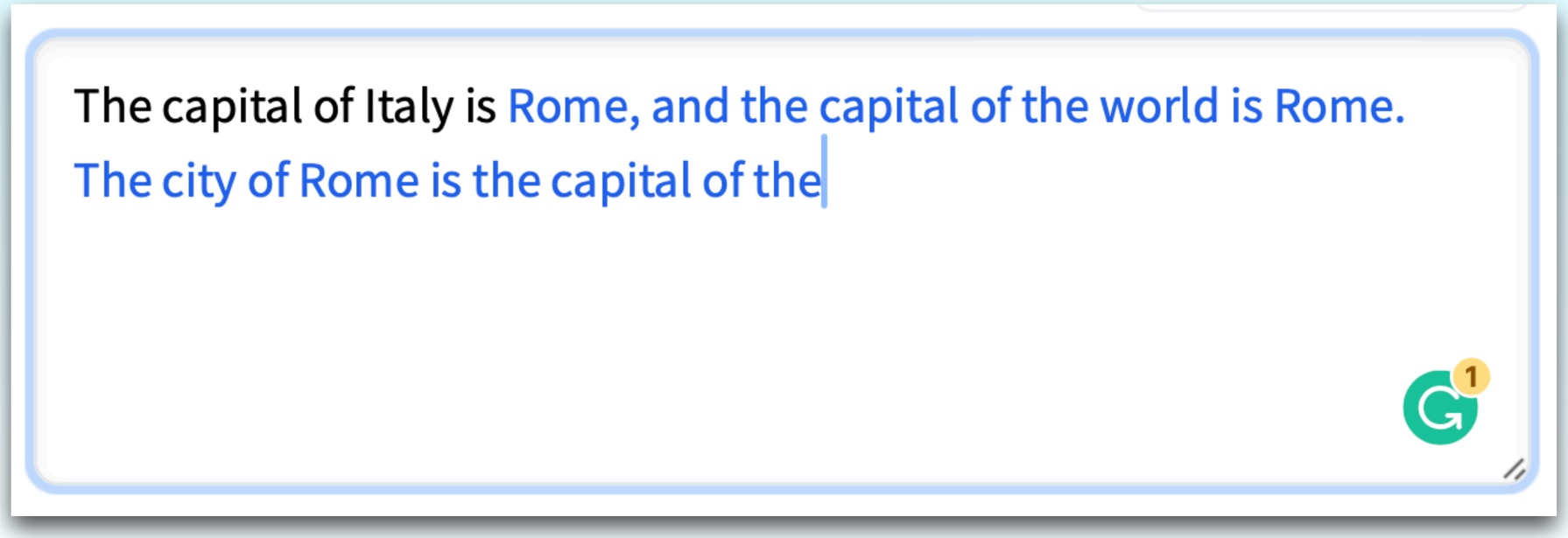
Why not each word as a token?

- “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

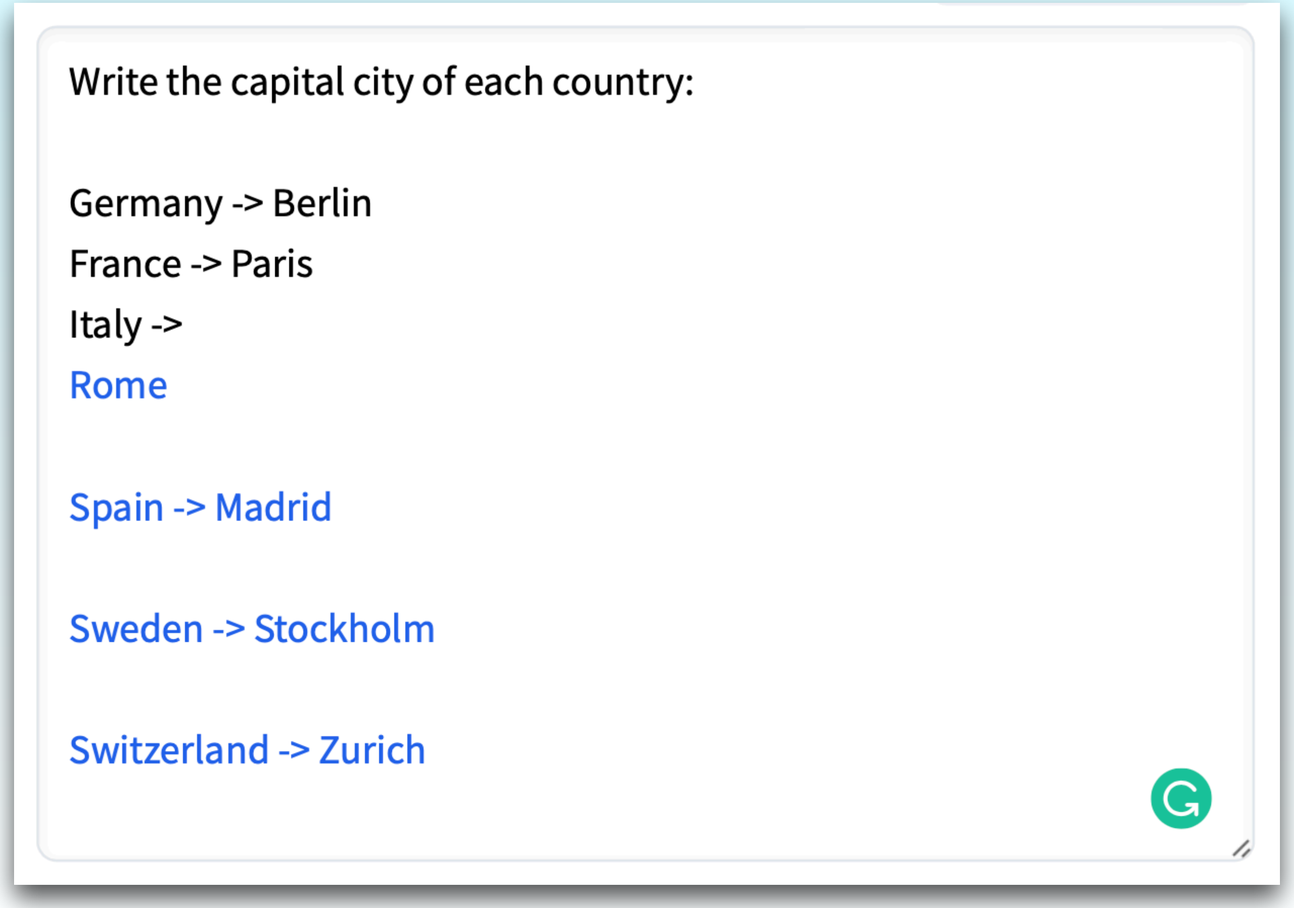
Zero-shot



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

Emergent Behavior

Few-shot (in-context examples)

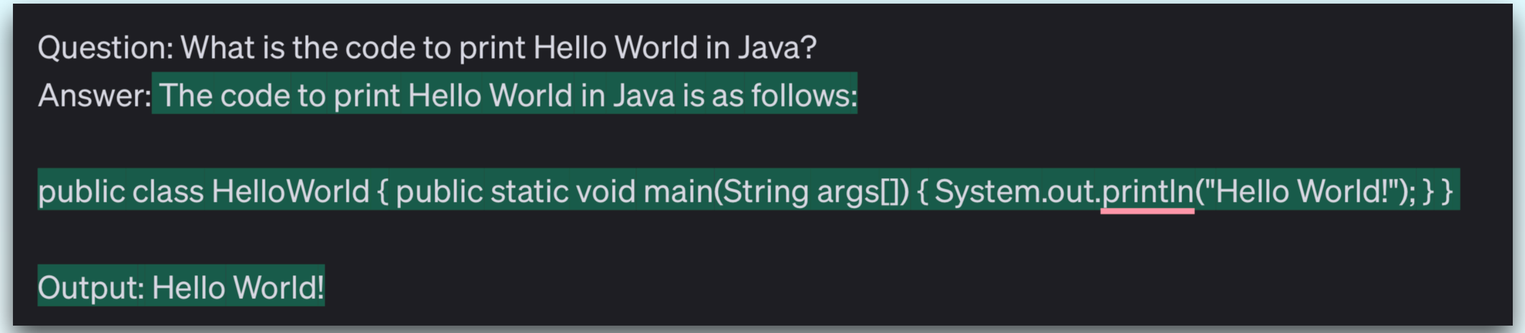


From https://huggingface.co/openai-community/gpt2

Emergent Behavior

The quality can differ specifically in more difficult questions

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



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

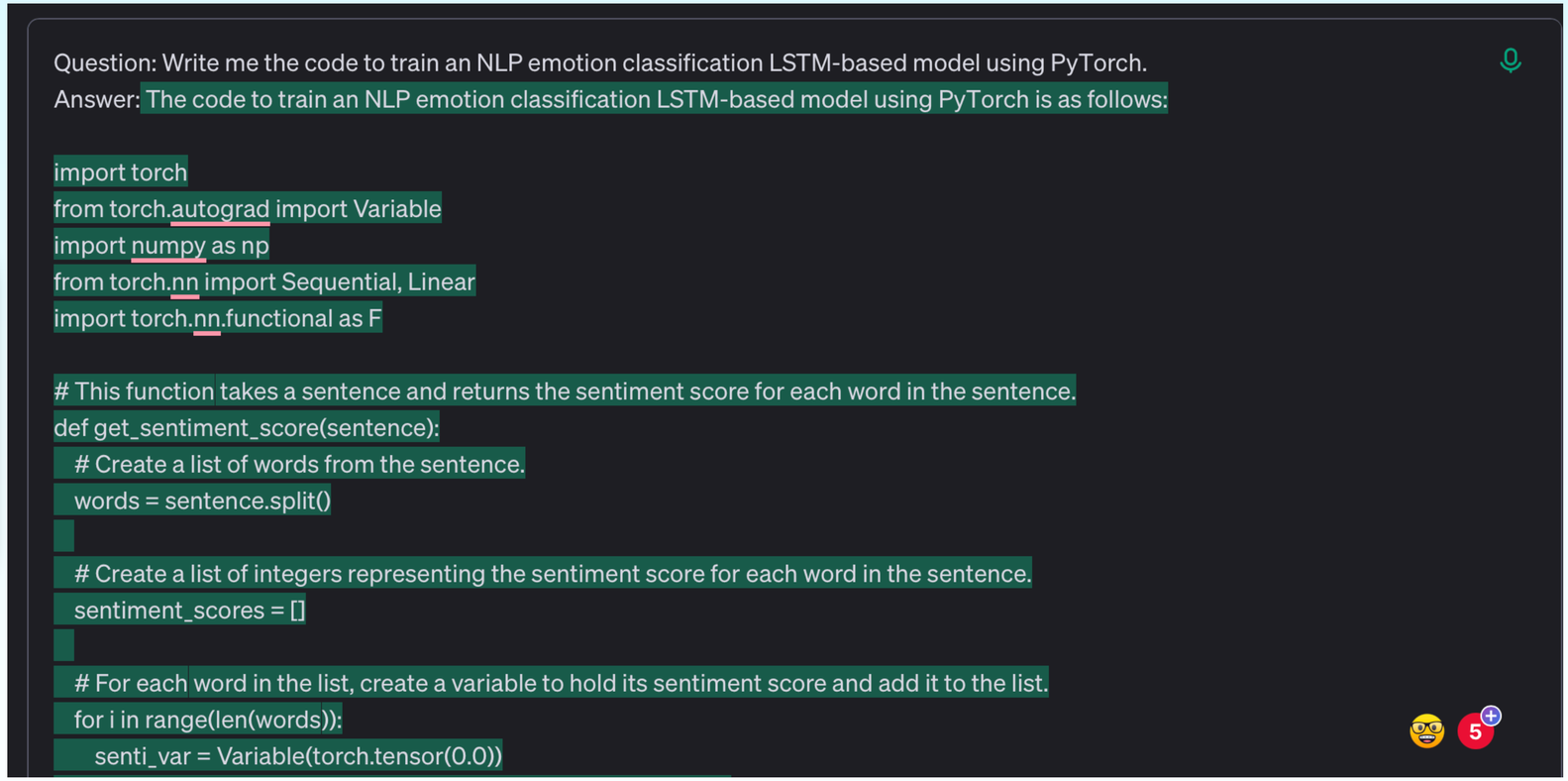
Emergent Behavior

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

- Prompt engineering

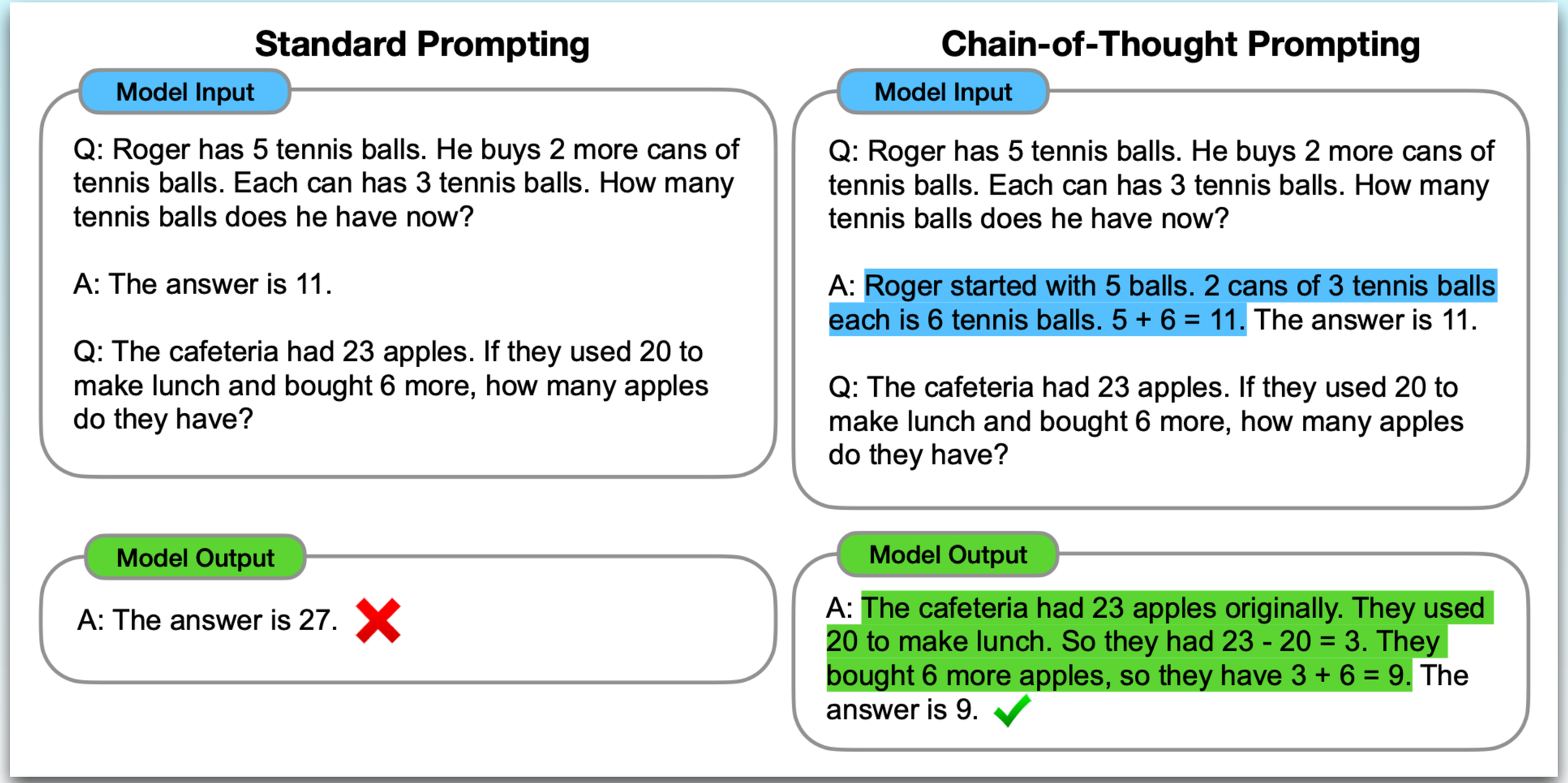
Question: Write me the code to train an NLP emotion classification LSTM-based model using PyTorch.
Answer:
I'm not sure what to call it. I'm not sure what to call it.
Answer

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



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

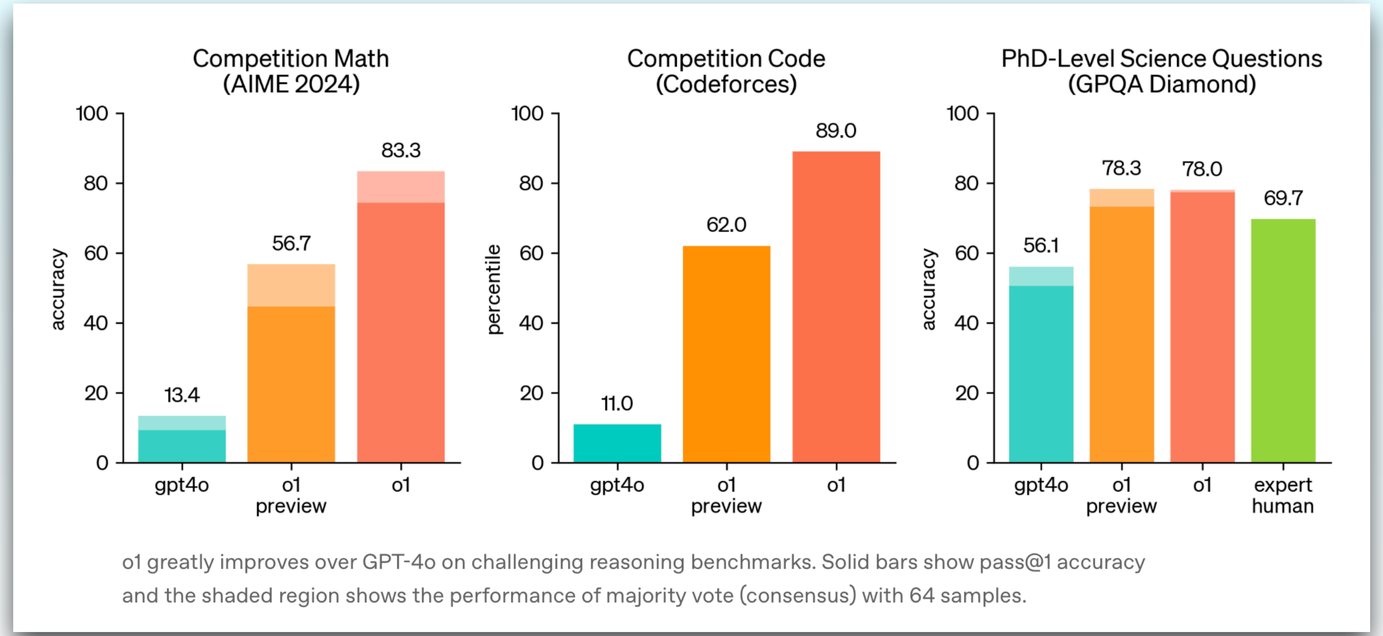
Another Emergent Behavior: Chain-of-Thought



Wei et al., 2022. “Chain-of-Thought Prompting Elicits Reasoning in Large Language Models”

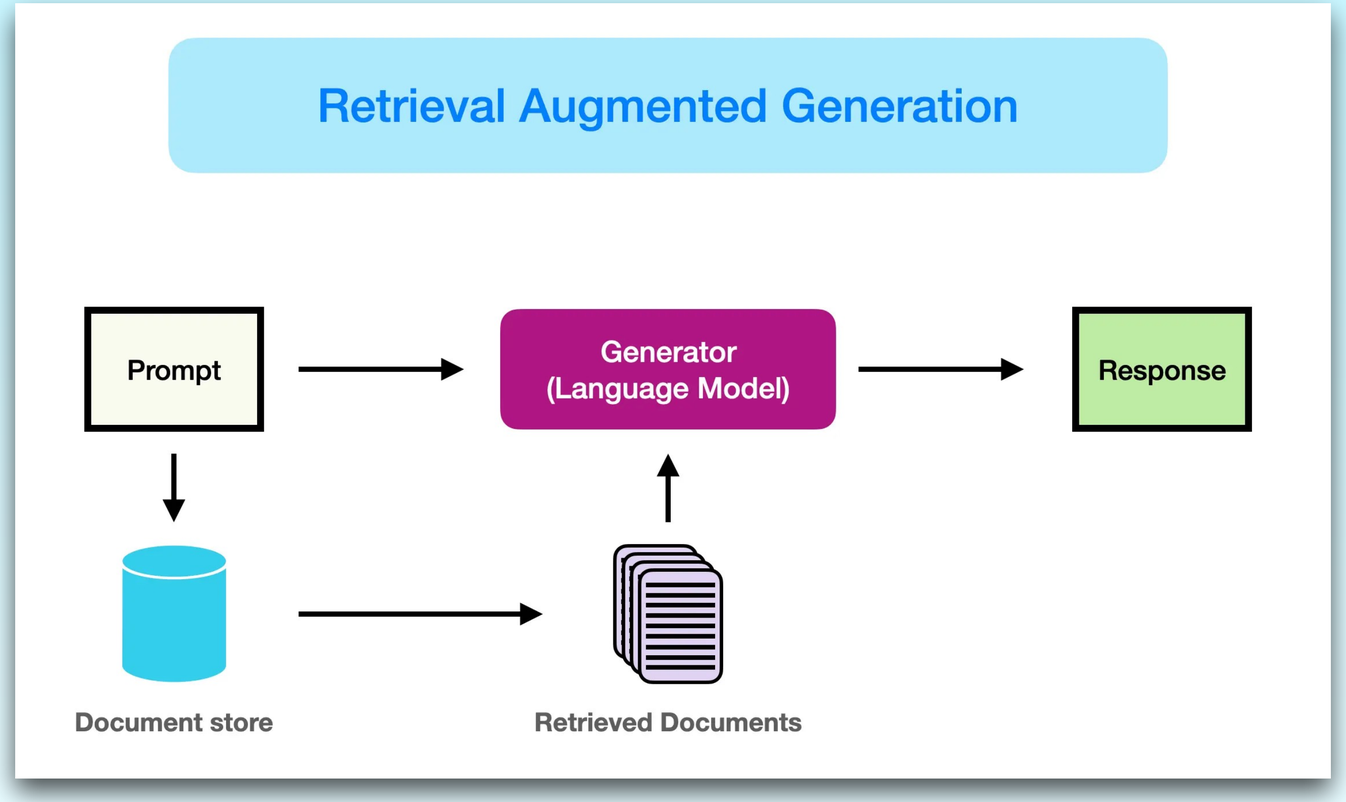
Reasoning Models

- (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



From https://openai.com/index/learning-to-reason-with-llms/

Retrieval Augmented Generation (RAG)



- 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

```python
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 i
    {"role": "user", "content": "Who are you?"},
]
outputs = pipe(
    messages,
    max_new_tokens=256,
)
print(outputs[0]["generated_text"][-1])
```

From https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct

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

```python
from openai import OpenAI
client = OpenAI()

response = client.responses.create(
    model="gpt-5",
    input="Write a one-sentence bedtime story about a unicorn."
)

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

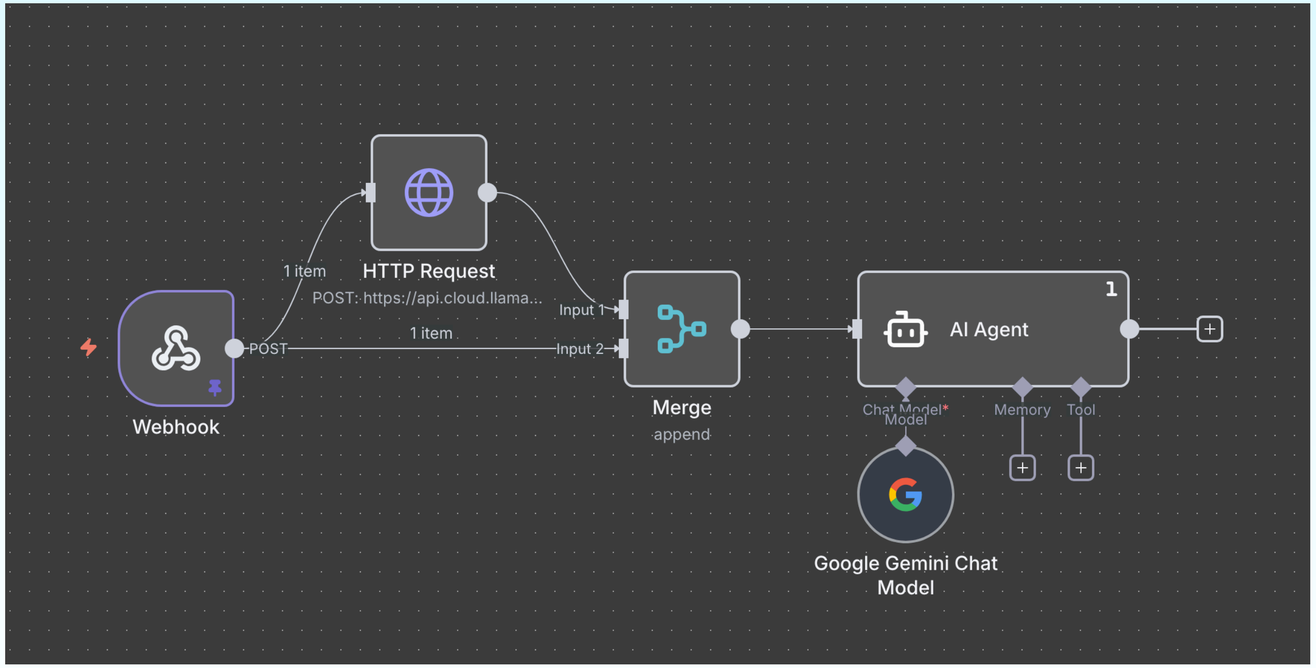
```python
from langchain.agents import create_agent

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"}]}
)
```

From https://docs.langchain.com/oss/python/langchain/quickstart

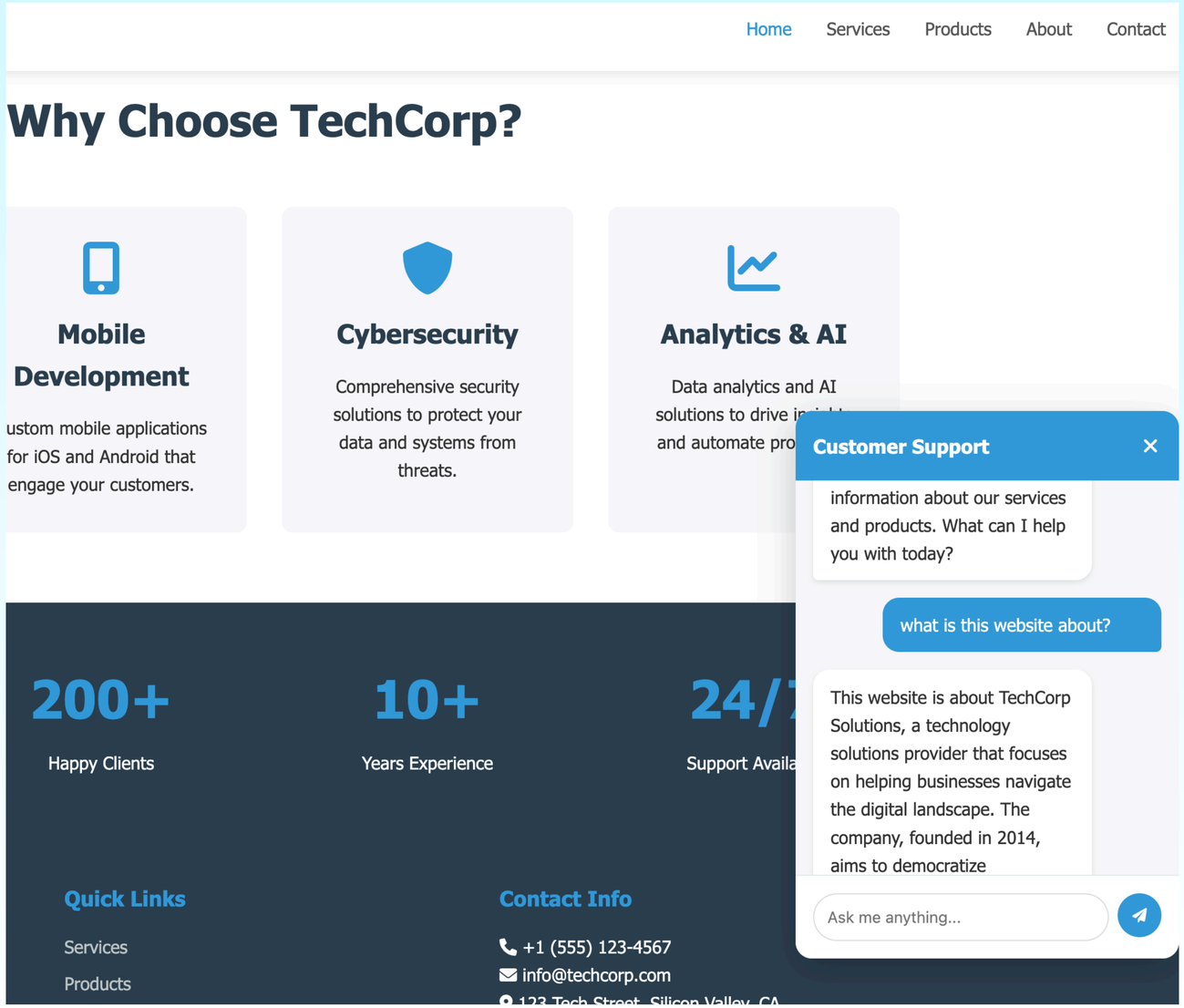
Chatbot assistant (implementation)

```python
from fastapi import FastAPI, Request
from fastapi.middleware.cors import CORSMiddleware
import uvicorn
from dotenv import load_dotenv
import os

app = FastAPI()

# # Add CORS middleware to handle cross-origin requests
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)


load_dotenv("real.env")
openai_key = os.getenv("OPENAI_API_KEY") # put your openai key here as a string. Example: "sk-proj-1234567890"
from openai import OpenAI
client = OpenAI(api_key=openai_key)

from llama_index.core import SimpleDirectoryReader, VectorStoreIndex
documents = SimpleDirectoryReader("frontend/static_pages/").load_data()
index = VectorStoreIndex.from_documents(documents, chunk_size=1000)
retriever = index.as_retriever(similarity_top_k=1)

@app.post("/chat")
async def process_chat_message(request: Request):
    try:
        data = await request.json()
        message = data.get("message")
        context= retriever.retrieve(message)
        response = client.chat.completions.create(model="gpt-4o-mini", messages=[{"role": "system", "content": f"You are a helpful
        {"role": "user", "content": message}])
        return {"response": response.choices[0].message.content}

    except Exception as e:
        print(f"Error in chat completion: {str(e)}")
        return {"response": f"Error: {str(e)}", "error": True}

if __name__ == '__main__':
    host = 'localhost'
    port = 8000
    uvicorn.run(app, host=host, port=port)
```

n8n



Chatbot assistant



Quiz time :)