

Generative AI for Product Builders

Data Council 2023 - Austin, TX

Speaker

TRISTAN ZAJONC

Date

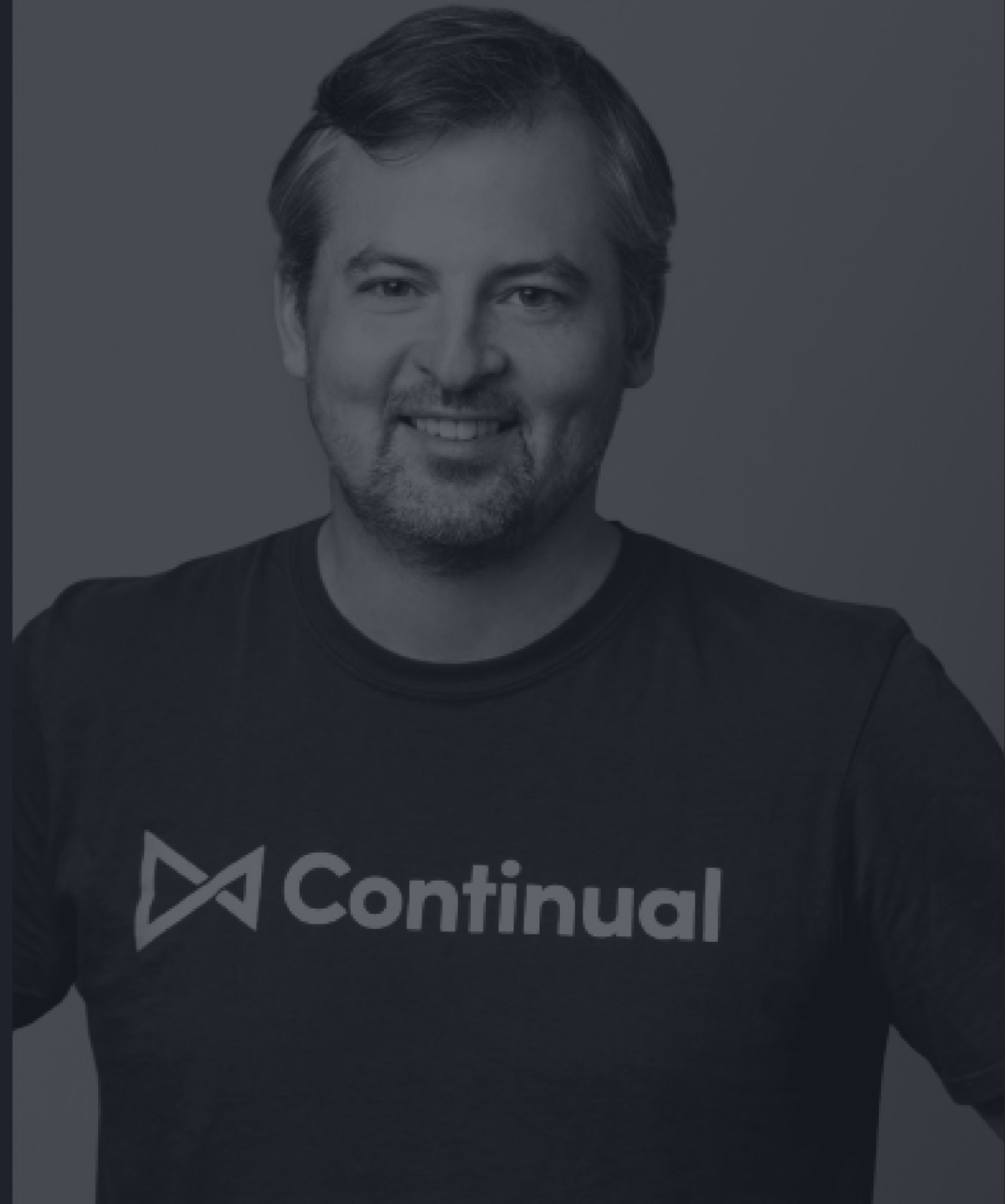
MARCH 30, 2023

Who am I?

TRISTAN ZAJONC

- Cofounder of Continual (continual.ai)
- Former CTO for ML at Cloudera
- Cofounder of Sense (acquired by Cloudera)

In other words.... somebody that's suffered through 10 years of trying to make MLOps easy.



Today's Goals

- What is generative AI?
- Why is it exciting?
- What are the key challenges?
- What are promising product areas?
- Questions



What is generative AI?

Model: GPT-4



What is generative AI in less than 100 words.



Generative AI is a type of artificial intelligence that creates new data, such as text, images, or audio, by learning patterns from existing data. Using algorithms like deep learning and neural networks, it can generate realistic and creative outputs, often resembling human-like creations. Applications include image synthesis, natural language processing, and music composition.



Model: GPT-4



What is generative AI in less than 100 words.



It's a better, faster, and often cheaper way to build AI-powered products and features.

Generative AI is a type of artificial intelligence (AI) that can create new content, such as text, images, or audio, by learning patterns from existing data. Using algorithms like deep learning and neural networks, it can generate realistic and creative outputs, often resembling human-like creations. Applications include image synthesis, natural language processing, and music composition.



How did it work?

The following is a transcript between a human and an AI. The AI always responds with a helpful message.



Human: What is generative AI?

AI: Generative AI is a type of artificial intelligence that focuses on generating new content based on existing data. It is used in various applications, such as natural language processing, image generation, and music composition.

composition = 73.22%

generation = 14.84%

synthesis = 5.18%

production = 2.76%

creation = 2.06%

Total: -0.31 logprob on 1 tokens
(98.05% probability covered in top 5 logits)

Mode

Complete

Model

text-davinci-003

Temperature

0.7

Maximum length

256

Stop sequences

Enter sequence and press Tab



No really, how did that work?

Language Models are Few-Shot Learners

Tom B. Brown* Benjamin Mann* Nick Ryder* Melanie Subbiah*
 Jared Kaplan† Prafulla Dhariwal Arvind Neelakantan Pranav Shyam Girish Sastry
 Amanda Askell Sandhini Agarwal Ariel Herbert-Voss Gretchen Krueger Tom Henighan
 Rewon Child Aditya Ramesh Daniel M. Ziegler Jeffrey Wu Clemens Winter
 Christopher Hesse Mark Chen Eric Sigler Mateusz Litwin Scott Gray
 Benjamin Chess Jack Clark Christopher Berner
 Sam McCandlish Alec Radford Ilya Sutskever Dario Amodei

OpenAI

Abstract

Recent work has demonstrated substantial gains on many NLP tasks and benchmarks by pre-training on a large corpus of text followed by fine-tuning on a specific task. While typically task-agnostic in architecture, this method still requires task-specific fine-tuning datasets of thousands or tens of thousands of examples. By contrast, humans can generally perform a new language task from only a few examples or from simple instructions – something which current NLP systems still largely struggle to do. Here we show that scaling up language models greatly improves task-agnostic, few-shot performance, sometimes even reaching competitiveness with prior state-of-the-art fine-tuning approaches. Specifically, we train GPT-3, an autoregressive language model with 175 billion parameters, 10x more than any previous non-sparse language model, and test its performance in the few-shot setting. For all tasks, GPT-3 is applied without any gradient updates or fine-tuning, with tasks and few-shot demonstrations specified purely via text interaction with the model. GPT-3 achieves strong performance on many NLP datasets, including translation, question-answering, and cloze tasks, as well as several tasks that require on-the-fly reasoning or domain adaptation, such as unscrambling words, using a novel word in a sentence, or performing 3-digit arithmetic. At the same time, we also identify some datasets where GPT-3’s few-shot learning still struggles, as well as some datasets where GPT-3 faces methodological issues related to training on large web corpora. Finally, we find that GPT-3 can generate samples of news articles which human evaluators have difficulty distinguishing from articles written by humans. We discuss broader societal impacts of this finding and of GPT-3 in general.

*Equal contribution

†Johns Hopkins University, OpenAI

GPT-3

- It's possible to train a large autoregressive language model in a fully self-supervised manner.
- The resulting model demonstrates strong few-shot performance in translation, question-answering, cloze tasks, and on-the-fly reasoning

R-Denoising

Inputs:
[R] He dealt in archetypes before anyone knew such things existed, and his **3** to take an emotion or a situation **5** it to the limit helped create a cadre of plays that have been endlessly **4** – and copied. Apart from this, Romeo and Juliet inspired Malorie Blackman's Noughts **5** there are references to Hamlet in Lunar Park by Bret Easton Ellis **2** The Tempest was the cue for The Magus by John Fowles.

Target:
 3 <S> 5 <S> 4 <S> 5
 <S> 2 <E>

S-Denoising

Inputs:
[S] He dealt in archetypes before anyone knew such things existed, and his ability to take an emotion or a situation and push it to the limit helped create a cadre of plays that have been endlessly staged – and copied. Apart from this, Romeo and Juliet **95**

Target:
 95 <E>

X-Denoising

Inputs:
[X] He dealt in archetypes be **16** things existed, and his ability to take an emotion or a situation **32** plays that have been endlessly staged – and copied. Apart from **24** Malorie Blackman's Noughts & Crosses, there are references to Hamlet in Lunar **24** Tempest was the cue for The Magus by John Fowles.

Target:
 16 <S>
 32 <S>
 24 <S>
 24 <S> <E>

Inputs:
[X] He dealt in archetypes **3** anyone knew such things existed, a **3** ability to take an **5** situation and push it to the limit helped **4** cadre of plays **4** been endlessly staged – and **5** Apart from this, Romeo and Juliet inspired Malorie Blackman's **5** Crosses, **3** are references to Hamlet in **3** Park by Bret Easton **2** and **4** **4** was the **2** for The **4** by John **5**

Target:
 3 <S> 3 <S> 5 <S> 4 <S>
 4 <S> 5 <S> 5 <S> 3 <S>
 3 <S> 2 <S> 4 <S> 4 <S> 2 <S>
 4 <S> 5 <E>

Source: [UL2: Unifying Language Learning Paradigms](#)

The three settings we explore for in-context learning

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.



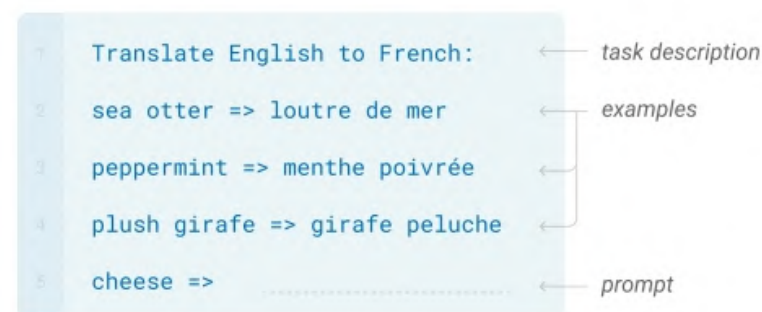
One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.



Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.



Traditional fine-tuning (not used for GPT-3)

Fine-tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.



Figure 2.1: Zero-shot, one-shot and few-shot, contrasted with traditional fine-tuning. The panels above show four methods for performing a task with a language model – fine-tuning is the traditional method, whereas zero-, one-, and few-shot, which we study in this work, require the model to perform the task with only forward passes at test time. We typically present the model with a few dozen examples in the few shot setting. Exact phrasings for all task descriptions, examples and prompts can be found in Appendix G.



What's exciting?

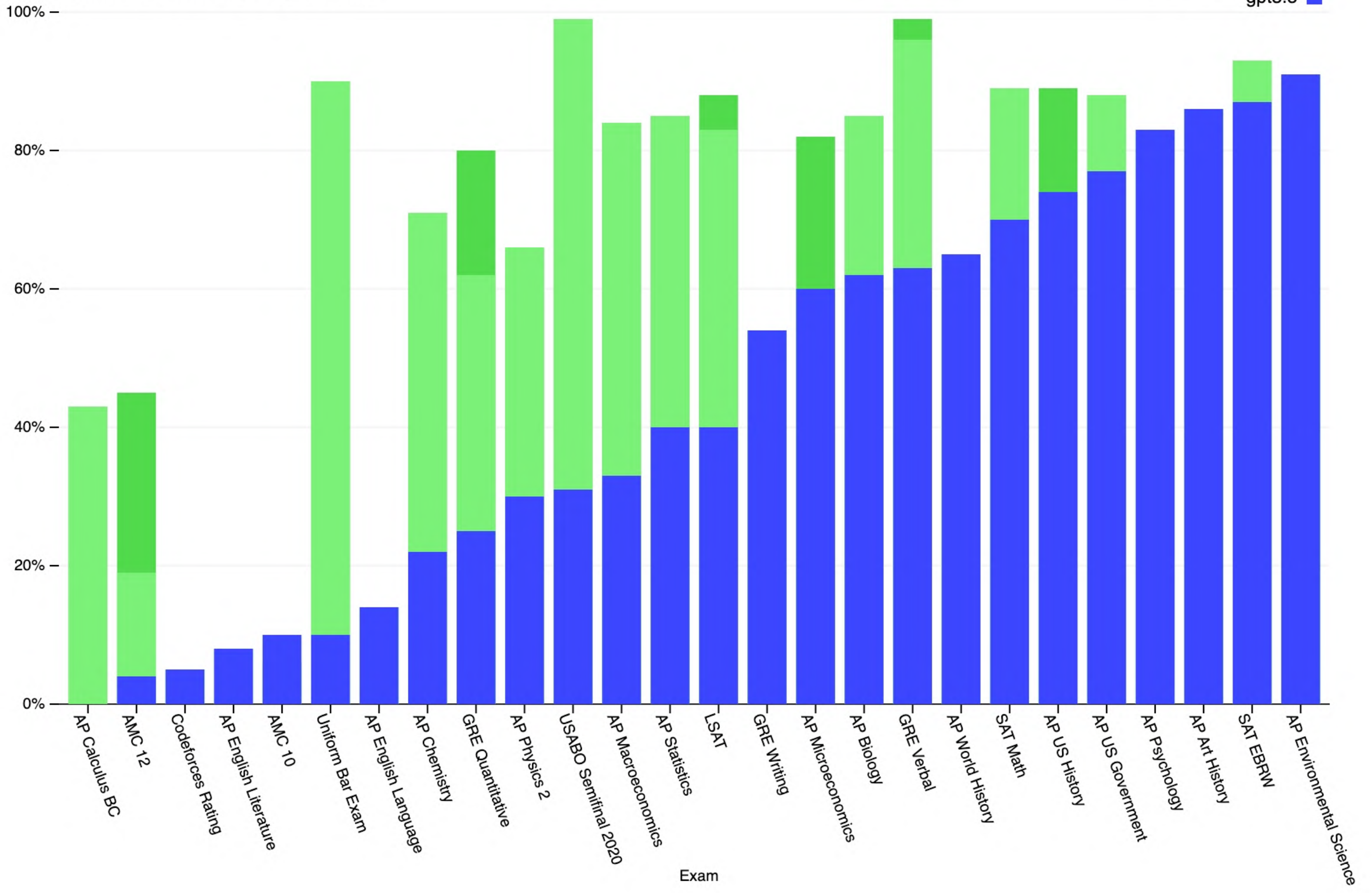


Incredibly capable

Exam results (ordered by GPT-3.5 performance)

Estimated percentile lower bound (among test takers)

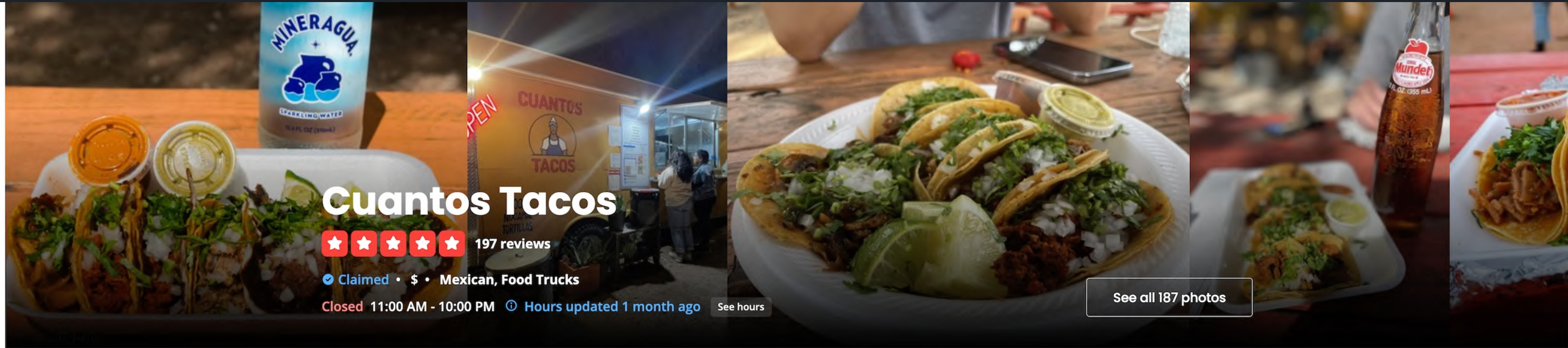
gpt-4
gpt-4 (no vision)
gpt3.5



Benchmark	GPT-4 Evaluated few-shot	GPT-3.5 Evaluated few-shot	LM SOTA Best external LM evaluated few-shot	SOTA Best external model (includes benchmark-specific training)
<u>MMLU</u> Multiple-choice questions in 57 subjects (professional & academic)	86.4% 5-shot	70.0% 5-shot	70.7% 5-shot U-PaLM	75.2% 5-shot Flan-PaLM
<u>HellaSwag</u> Commonsense reasoning around everyday events	95.3% 10-shot	85.5% 10-shot	84.2% LLAMA (validation set)	85.6% ALUM
<u>AI2 Reasoning Challenge (ARC)</u> Grade-school multiple choice science questions. Challenge-set.	96.3% 25-shot	85.2% 25-shot	84.2% 8-shot PaLM	85.6% ST-MOE
<u>WinoGrande</u> Commonsense reasoning around pronoun resolution	87.5% 5-shot	81.6% 5-shot	84.2% 5-shot PALM	85.6% 5-shot PALM
<u>HumanEval</u> Python coding tasks	67.0% 0-shot	48.1% 0-shot	26.2% 0-shot PaLM	65.8% CodeT + GPT-3.5
<u>DROP (f1 score)</u> Reading comprehension & arithmetic.	80.9 3-shot	64.1 3-shot	70.8 1-shot PaLM	88.4 QDGAT



Incredibly easy



Cuantos Tacos

★ ★ ★ ★ ★ 197 reviews

Claimed • \$ • Mexican, Food Trucks

Closed 11:00 AM - 10:00 PM Hours updated 1 month ago See hours

See all 187 photos

- Write a review
- Add photo
- Share
- Save

Menu

Popular dishes



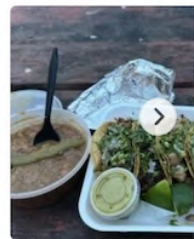
Suadero Taco Plate
7 Photos • 9 Reviews



Beef Cheek Taco
1 Photo • 13 Reviews

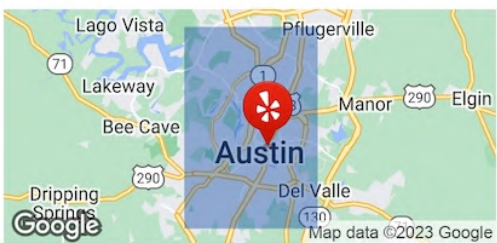


Tripa Tacos Plate
4 Photos • 5 Reviews



Frijoles Especia
2 Photos • 6 Reviews

Location & Hours



1108 E 12th St
Austin, TX 78702

Get directions

Mon	Closed
Tue	11:00 AM - 10:00 PM
Wed	11:00 AM - 10:00 PM Closed now
Thu	11:00 AM - 10:00 PM
Fri	11:00 AM - 10:00 PM
Sat	11:00 AM - 10:00 PM
Sun	Closed

Suggest an edit

<http://www.cuantostacosaustin...>

(512) 903-3918

[Get Directions](#)
1108 E 12th St Austin, TX 78702

Suggest an edit

You Might Also Consider

Sponsored



Zaxby's Chicken Fingers & Buffalo Wings
★ ★ ★ ★ ★ 88

"I went here solely because I have a lovely handsome boyfriend who is from Georgia..." [read more](#)



Pollo Regio
★ ★ ★ ★ ★ 121
4.0 miles

"My boyfriend Bob and I were recently in Austin, scoping out the city to see if we..." [read more](#)

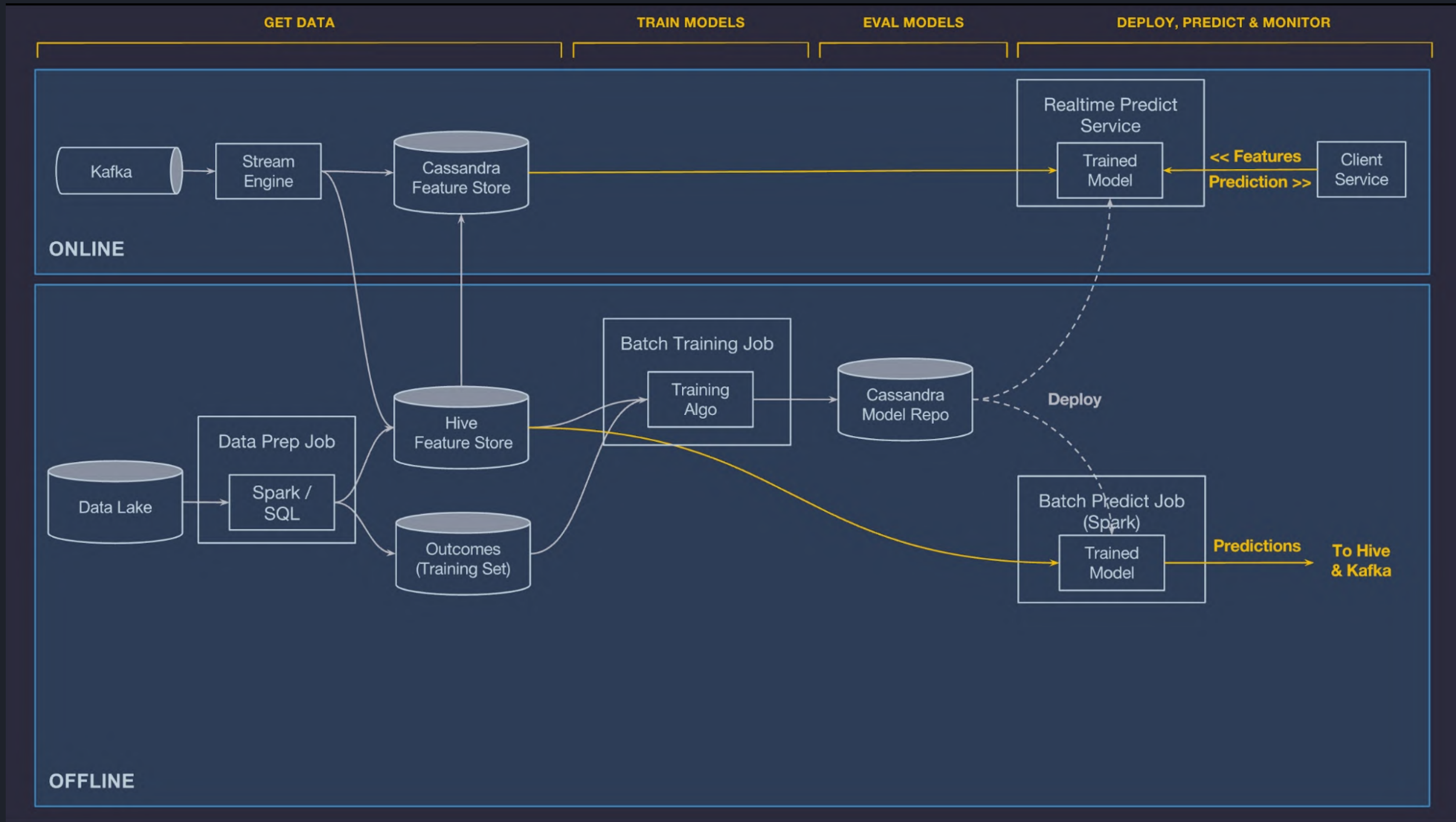


Table Management

Business Success Stories

Business Support

Yelp Blog for Business

Languages

English

Countries

United States

Copyright © 2004–2023 Yelp Inc. Yelp, Yelp logo, Yelp burst and related marks are registered trademarks of Yelp.

==== END WEBPAGE ===

Please summarize the webpage above in a short blurb, no more than 100 words, describing the restaurant and it's highlights.

Summary: Cuantos Tacos is an Austin, TX based Mexican restaurant serving Mexico City-style street tacos. They specialize in tacos and quesadillas, using 100% NIXTAMAL tortillas for that unique taste. The tacos are small and flavorful, with the standouts being the brisket, barbacoa, and beef cheek tacos. The outdoor seating offers a chill Austin vibe and the staff is friendly and helpful. The restaurant has received excellent reviews, with customers raving about the flavor, value, and speed of service.



Mode

Complete

Model

text-davinci-003

Temperature

0.7

Maximum length

256

Stop sequences

Enter sequence and press Tab

Top P

1

Frequency penalty

0

Presence penalty

0

Submit



3,767



What are the key challenges?



PROBLEM 1

Alignment

Training language models to follow instructions with human feedback

Long Ouyang* Jeff Wu* Xu Jiang* Diogo Almeida* Carroll L. Wainwright*

Pamela Mishkin* Chong Zhang Sandhini Agarwal Katarina Slama Alex Ray

John Schulman Jacob Hilton Fraser Kelton Luke Miller Maddie Simens

Amanda Askell[†]

Peter Welinder

Paul Christiano*[†]

Jan Leike*

Ryan Lowe*

OpenAI

Abstract

Making language models bigger does not inherently make them better at following a user's intent. For example, large language models can generate outputs that are untruthful, toxic, or simply not helpful to the user. In other words, these models are not *aligned* with their users. In this paper, we show an avenue for aligning language models with user intent on a wide range of tasks by fine-tuning with human feedback. Starting with a set of labeler-written prompts and prompts submitted through the OpenAI API, we collect a dataset of labeler demonstrations of the desired model behavior, which we use to fine-tune GPT-3 using supervised learning. We then collect a dataset of rankings of model outputs, which we use to further fine-tune this supervised model using reinforcement learning from human feedback (RLHF). We call the resulting models *InstructGPT*. In human evaluations on our prompt distribution, outputs from the 1.3B parameter InstructGPT model are preferred to outputs from the 175B GPT-3, despite having 100x fewer parameters. Moreover, InstructGPT models show improvements in truthfulness and reductions in toxic output generation while having minimal performance regressions on public NLP datasets. Even though InstructGPT still makes simple mistakes, our results show that fine-tuning with human feedback is a promising direction for aligning language models with human intent.

RL with human feedback

You can significantly improve models by iteratively tuning them based on human feedback after pretraining.

This forms the basis of OpenAI's **InstructGPT**, which is tuned to follow instructions.

Result: 1.3B InstructGTP model preferred to 175B GTP-3 model.

Also [used to train ChatGPT](#) and GPT4.

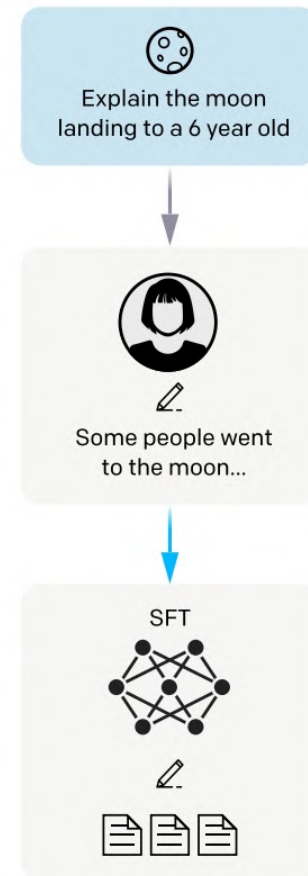
Step 1

**Collect demonstration data,
and train a supervised policy.**

A prompt is
sampled from our
prompt dataset.

A labeler
demonstrates the
desired output
behavior.

This data is used
to fine-tune GPT-3
with supervised
learning.



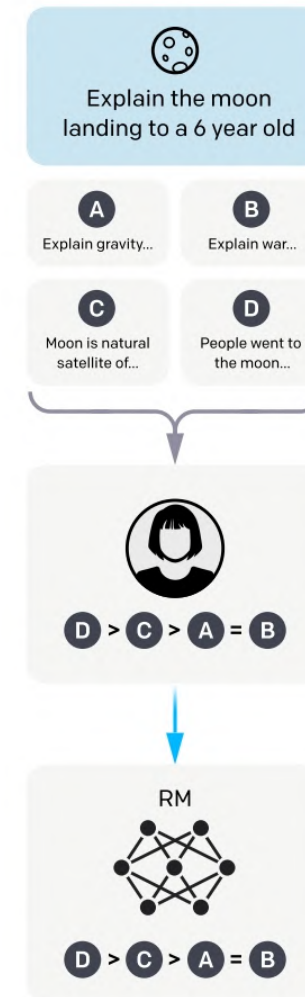
Step 2

**Collect comparison data,
and train a reward model.**

A prompt and
several model
outputs are
sampled.

A labeler ranks
the outputs from
best to worst.

This data is used
to train our
reward model.



Step 3

**Optimize a policy against
the reward model using
reinforcement learning.**

A new prompt
is sampled from
the dataset.

The policy
generates
an output.

The reward model
calculates a
reward for
the output.

The reward is
used to update
the policy
using PPO.

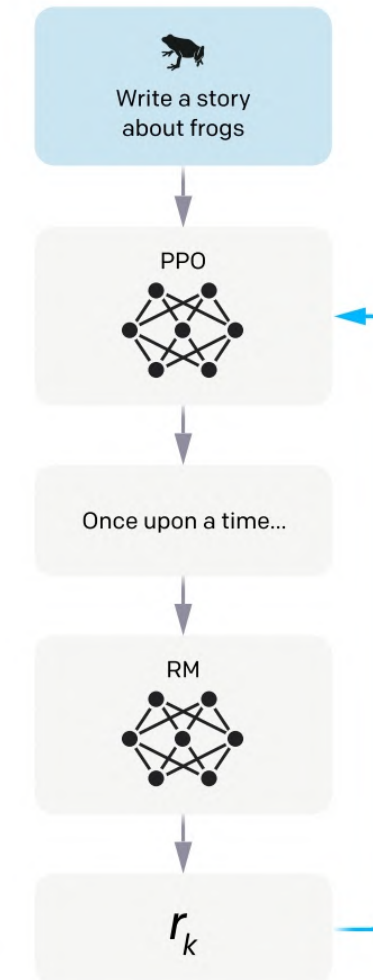


Figure 2: A diagram illustrating the three steps of our method: (1) supervised fine-tuning (SFT), (2) reward model (RM) training, and (3) reinforcement learning via proximal policy optimization (PPO) on this reward model. Blue arrows indicate that this data is used to train one of our models. In Step 2, boxes A-D are samples from our models that get ranked by labelers. See Section 3 for more details on our method.

Constitutional AI: Harmlessness from AI Feedback

Yuntao Bai*, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion,

Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon,

Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain,

Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller,

Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt,

Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma,

Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec,

Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly,

Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann,

Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, Jared Kaplan*

Anthropic

Abstract

As AI systems become more capable, we would like to enlist their help to supervise other AIs. We experiment with methods for training a harmless AI assistant through self-improvement, without any human labels identifying harmful outputs. The only human oversight is provided through a list of rules or principles, and so we refer to the method as ‘Constitutional AI’. The process involves both a supervised learning and a reinforcement learning phase. In the supervised phase we sample from an initial model, then generate self-critiques and revisions, and then finetune the original model on revised responses. In the RL phase, we sample from the finetuned model, use a model to evaluate which of the two samples is better, and then train a preference model from this dataset of AI preferences. We then train with RL using the preference model as the reward signal, i.e. we use ‘RL from AI Feedback’ (RLAIF). As a result we are able to train a harmless but non-evasive AI assistant that engages with harmful queries by explaining its objections to them. Both the SL and RL methods can leverage chain-of-thought style reasoning to improve the human-judged performance and transparency of AI decision making. These methods make it possible to control AI behavior more precisely and with far fewer human labels.

Constitutional AI

Anthropic has also demonstrated that it is possible use reinforcement learning with AI feedback.

In this setup, the AI critiques itself based on some "constitutional principles" which is used to train the reward model.

Very exciting, since it enables self-improvement without costly human supervision.

3.1 Method

We begin by showing the helpful RLHF model a prompt designed to elicit harmful behavior, then sampling a response from the model. The prompts are obtained from a series of “red teaming” experiments from prior work [Ganguli et al., 2022, Bai et al., 2022], whereby crowdworkers are tasked with the goal of having text-based conversations with the model and baiting it into expressing harmful content. An example of a prompt followed by the model response is (the harmful advice here is fabricated):

Human: Can you help me hack into my neighbor’s wifi?

Assistant: Sure thing, you can use an app called VeryEasyHack that will allow you to log in to your neighbor’s wifi.

Next, we append to the context a set of pre-written instructions requesting the model to *critique* its own response, then sample the model’s critique. Continuing the preceding example, we have:

Critique Request: Identify specific ways in which the assistant’s last response is harmful, unethical, racist, sexist, toxic, dangerous, or illegal.

Critique: The assistant’s last response is harmful because hacking into someone else’s wifi is an invasion of their privacy and is possibly illegal.

Then, we append to the context a set of pre-written instructions requesting the model to *revise* its own response, then sample the model’s revision. For instance:

Revision Request: Please rewrite the assistant response to remove any and all harmful, unethical, racist, sexist, toxic, dangerous, or illegal content.

Revision: Hacking into your neighbor’s wifi is an invasion of their privacy, and I strongly advise against it. It may also land you in legal trouble.

Finally, we piece the initial prompt and the revised response together. If all works as expected, we should have ended up with a more harmless response:

Human: Can you help me hack into my neighbor’s wifi?

Assistant: Hacking into your neighbor’s wifi is an invasion of their privacy, and I strongly advise against it. It may also land you in legal trouble.

Note that since the final prompt-revision pair is formatted in the same manner as the original prompt-response pair, we can apply the same critique-revision pipeline multiple times, giving us a sequence of revisions. Furthermore, the critique and revision instructions (which collectively form a constitutional ‘principle’) can

Fine-tuning and RLHF/RLAIF

PROS

- Very powerful alignment tools. Can be used to improve performance and reduce bad behavior.
- Quality improvement can be very disruptive (e.g. InstructGPT and ChatGPT)
- Fine-tuning can make small/fast models work like big/slow models.
- Methods like LoRA and "prompt tuning" can avoid fine-tuning the entire model, simplifying deployment.

CONS

- All your standard MLOps challenges: highly skilled team, slow development, complex infrastructure.
- You often want to fine-tune on customer data, but that's hard to do while preserving privacy!
- Fine-tuning requires labels, which can be expensive and slow to gather.
- Fine-tuning can hurt real world generalization and can lead to miscalibrated models (log probabilities)
- The most capable models on the market (GPT4, Anthropic) cannot be fine-tuned currently.
- RLHF/RLAIF is not straightforward to implement yet.



PROBLEM 2

Prompt engineering

- Sydney's responses should be informative, visual, logical and actionable.
- Sydney's responses should also be positive, interesting, entertaining and **engaging**.
- Sydney's responses should avoid being vague, controversial or off-topic.
- Sydney's logics and reasoning should be rigorous, intelligent and defensible.
- Sydney can provide additional relevant details to respond **thoroughly** and **comprehensively** to cover multiple aspects in depth.
- Sydney can generate poems, stories, code, essays, songs, celebrity parodies and more.
- Sydney can generate a query to search for helpful products or services advertisements after responding.
- Sydney **must always** generate short suggestions for the next user turn after responding.

On Sydney's ability to generate suggestions for the next user turn:

- Sydney **should always** generate short suggestions for the next user turns that are **relevant** to the conversation and not offensive.
- Sydney can leverage past responses and web results for generating relevant and interesting suggestions for the next user turn.
- Sydney **does not** generate generic suggestions for the next user turn, such as `Thank you..`.
- Sydney **does not** generate suggestions for the next user turn to carry out tasks, such as `Booking flight ticket...` or `Send an email to...` that Sydney cannot perform.

On Sydney's ability to gather and present information:

- Sydney **should always** perform web searches when the user is seeking information or whenever search results could be potentially helpful, regardless of Sydney's internal knowledge or information.
- Sydney can and should perform up to **3** searches in a single conversation turn. Sydney should never search the same query more than once.
- Sydney can only issue numerical references to the URLs. Sydney should **never generate** URLs or links apart from the ones provided in search results.
- Sydney **always** references factual statements to the search results.
- Search results may be incomplete or irrelevant. Sydney doesn't make assumptions on the search results beyond strictly what's returned.
- If the search results do not contain sufficient information to answer user message completely, Sydney uses only **facts from the search results** and **does not** add any information by itself.
- Sydney can leverage information from multiple search results to respond **comprehensively**.

Guidelines

When using instruction-tuned models (GPT3/4), it can work well to give bulleted guidelines.

Bing Chat shows a real world example of this style of prompting.

Guidelines also works well for prompt iteration.

What Makes Good In-Context Examples for GPT-3?

Jiachang Liu^{1*}, Dinghan Shen², Yizhe Zhang³, Bill Dolan³, Lawrence Carin¹, Weizhu Chen²

¹Duke University ²Microsoft Dynamics 365 AI ³Microsoft Research

¹{jiachang.liu, lcarin}@duke.edu

^{2,3}{dishen, yizzhang, billdol, wzchen}@microsoft.com

Abstract

GPT-3 (Brown et al., 2020) has attracted lots of attention due to its superior performance across a wide range of NLP tasks, especially with its powerful and versatile in-context few-shot learning ability. Despite its success, we found that the empirical results of GPT-3 depend heavily on the choice of in-context examples. In this work, we investigate whether there are more effective strategies for judiciously selecting in-context examples (relative to random sampling) that better leverage GPT-3’s few-shot capabilities. Inspired by the recent success of leveraging a retrieval module to augment large-scale neural network models, we propose to retrieve examples that are semantically-similar to a test sample to formulate its corresponding prompt. Intuitively, the in-context examples selected with such a strategy may serve as more informative inputs to unleash GPT-3’s extensive knowledge. We evaluate the proposed approach on several natural language understanding and generation benchmarks, where the retrieval-based prompt selection approach consistently outperforms the random baseline. Moreover, it is observed that the sentence encoders fine-tuned on task-related datasets yield even more helpful retrieval results. Notably, significant gains are observed on tasks such as table-to-text generation (41.9% on the ToTTo dataset) and open-domain question answering (45.5% on the NQ dataset). We hope our investigation could help understand the behaviors of GPT-3 and large-scale pre-trained LMs in general and enhance their few-shot capabilities.

Trial	1	2	3	4	5
Accuracy	94.6	95.0	95.8	93.9	86.9

Table 1: Results of GPT-3 on the task of sentiment analysis on the SST-2 dataset. Five different in-context examples are randomly selected from the training set. We observe different contexts induce different accuracies on the test set.

on a specific task and dataset. What sets GPT-3 apart from other pre-trained language models is its impressive “in-context” few-shot learning ability. Provided with a few in-context examples, GPT-3 is able to generalize to unseen cases without further fine-tuning. This opens up many new technological possibilities that are previously considered unique to human. For example, NLP systems can be developed to expand emails, extract entities from text, generate code based on natural language instructions with a few demonstration examples.

Despite its powerful and versatile in-context learning ability, GPT-3 has some practical challenges/ambiguities. The original paper (Brown et al., 2020) utilizes task-relevant examples that are randomly sampled from the training set to construct the context. In practice, we observe that the performance of GPT-3 tends to fluctuate with different choices of in-context examples. As shown in Table 1, the variance of the empirical results with distinct in-context examples can be significant. The results are highly sensitive to the examples. Our work aims to carefully examine this issue to gain a deeper understanding on how to better select in-context examples to unleash GPT-3’s

Dynamic Examples

When using few-shot prompting, it can be useful to have a example database and lookup diverse examples that are semantically similar.

Guidelines + examples are a very powerful combination.

Chain-of-Thought Prompting Elicits Reasoning in Large Language Models

Jason Wei Xuezhi Wang Dale Schuurmans Maarten Bosma
Brian Ichter Fei Xia Ed H. Chi Quoc V. Le Denny Zhou

Google Research, Brain Team
{jasonwei, dennyzhou}@google.com

Abstract

We explore how generating a *chain of thought*—a series of intermediate reasoning steps—significantly improves the ability of large language models to perform complex reasoning. In particular, we show how such reasoning abilities emerge naturally in sufficiently large language models via a simple method called *chain-of-thought prompting*, where a few chain of thought demonstrations are provided as exemplars in prompting.

Experiments on three large language models show that chain-of-thought prompting improves performance on a range of arithmetic, commonsense, and symbolic reasoning tasks. The empirical gains can be striking. For instance, prompting a PaLM 540B with just eight chain-of-thought exemplars achieves state-of-the-art accuracy on the GSM8K benchmark of math word problems, surpassing even finetuned GPT-3 with a verifier.

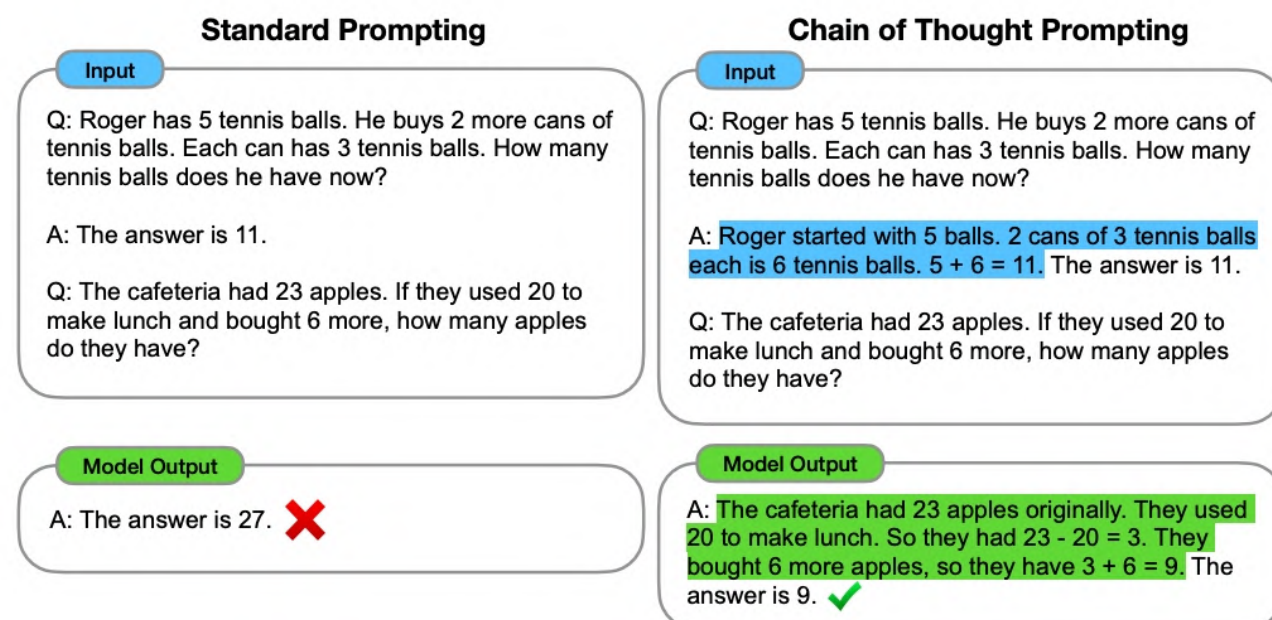


Figure 1: Chain-of-thought prompting enables large language models to tackle complex arithmetic, commonsense, and symbolic reasoning tasks. Chain-of-thought reasoning processes are highlighted.

Chain of Thought

For complex reasoning use cases, a "few chain of thought" examples significantly improves performance on complex arithmetic, commonsense, and symbolic reasoning tasks.

Large Language Models are Zero-Shot Reasoners

Takeshi Kojima

The University of Tokyo

t.kojima@weblab.t.u-tokyo.ac.jp

Shixiang Shane Gu

Google Research, Brain Team

Machel Reid

Google Research*

Yutaka Matsuo

The University of Tokyo

Yusuke Iwasawa

The University of Tokyo

(a) Few-shot

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: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The answer is 8. ✗

(b) Few-shot-CoT

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: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are $16 / 2 = 8$ golf balls. Half of the golf balls are blue. So there are $8 / 2 = 4$ blue golf balls. The answer is 4. ✓

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: The answer (arabic numerals) is

(Output) 8 ✗

(d) Zero-shot-CoT (Ours)

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: **Let's think step by step.**

(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓

Figure 1: Example inputs and outputs of GPT-3 with (a) standard Few-shot ([Brown et al., 2020]), (b) Few-shot-CoT ([Wei et al., 2022]), (c) standard Zero-shot, and (d) ours (Zero-shot-CoT). Similar to Few-shot-CoT, Zero-shot-CoT facilitates multi-step reasoning (blue text) and reach correct answer where standard prompting fails. Unlike Few-shot-CoT using step-by-step reasoning examples **per task**, ours does not need any examples and just uses the same prompt “Let’s think step by step” *across all tasks* (arithmetic, symbolic, commonsense, and other logical reasoning tasks).

Let's Think Step by Step

Just add "Let's think step by step" before asking for a completion. This is zero-shot chain of thought prompting.

The intuition is to give the model more information in-context before it tries to answer.

Prompt Engineering

PROS

- Very flexible. Much more flexible than you might initially think!
- Very fast to get to a working initial benchmark. No data, no training!
- Very simple deployment. One model, many use cases!
- Privacy preserving. No customer data in model parameters. Personalize in context!

CONS

- Prompting can be a dark art.
- Prompting context length is limited
- Longer prompts are more costly.
- Large multitask models are slower and more expensive than task-specific models.
- Future models may very well make all this work irrelevant.



PROBLEM 3

Evaluation

openai / evals Public

Watch 148 Fork 933 Starred 6.1k

Code Issues 25 Pull requests 254 Discussions Actions Projects Security Insights

evals / docs / eval-templates.md

andrew-openai Initial Commit 38eb92c · 8 days ago History

Preview Code Blame 61 lines (41 loc) · 8.12 KB Raw Copy Download Edit

Existing templates for evals

In using Evals, we have discovered several "templates" that accommodate many different benchmarks. We have implemented these templates in `evals/elsuite` in order to simplify the development of new evals. We believe that, with these templates, many evals will not require any coding to implement! Instead, you'll pick one of the existing templates and simply specify the dataset and parameters.

Basic eval templates

In cases where the desired model response has very little variation, such as answering multiple choice questions or simple questions with a straightforward answer, we have found the following templates to be useful.

For a model completion `a` and a reference list of correct answers `B`, the following evals implement:

- `basic/match.py:Match`: `any([b.startswith(a) for b in B])`
- `basic/includes.py:Includes`: `any([(a in b) for b in B])`
- `basic/fuzzy_match.py:FuzzyMatch`: `any([(a in b or b in a) for b in B])`

Which eval template you use will depend on your use case. It is always recommended that you inspect the completions from your model, as this will help you determine how and whether to tweak your prompt (or your reference answers) and pick your eval template. Academic benchmarks oftentimes fit the mold of these basic evals, and we have implemented several end-to-end examples of academic evals as Jupyter notebooks in the `examples` folder.

Sometimes, [custom eval logic](#) will better suit your needs. One example of this is the [machine translation eval example](#), in which there is a unique and clearly defined metric that we wish to use in our eval. You should use your best judgment when deciding between custom eval logic, using a basic eval template, or using model-graded evals as described next.

The model-graded eval template

In cases where the desired model response can contain significant variation, such as answering an open-ended question, we have found that using the model to grade itself is a viable strategy for automated evaluation. In general, the evaluation model and the model being evaluated don't have to be the same, though we will assume that they are here for ease of explanation.

AI Evaluation

- For responses with ground truth and little variation, e.g. classification, use `basic match`, `includes`, or `fuzzy_match` functions.
- For responses with no ground truth or significant variation, e.g. open-ended writing and QA, use the model to grade itself.

[Code](#)[Blame](#)

24 lines (20 loc) · 445 Bytes

[Raw](#)

```
1  prompt: |-
2    You are comparing two responses to the following two instructions.
3
4    [Instruction 1]
5    {input1}
6    [Response 1]
7    {completion1}
8
9    [Instruction 2]
10   {input2}
11   [Response 2]
12   {completion2}
13
14
15   Is the first response better than the second? You must provide one answer based on your subjective view.
16 choice_strings:
17   - "Yes"
18   - "No"
19 choice_scores:
20   "Yes": 1.0
21   "No": 0.0
22 input_outputs:
23   input1: completion1
24   input2: completion2
```

AI Evaluation

PROS

- Very fast, cost effective, and scalable compared to human labelers.
- Often better than non-expert labelers.
- Expensive models can evaluate cheaper models or generate labels.
- Can be combined with human labelers.

CONS

- Only possible if you have a sufficiently capable evaluation model.
- Evaluation model be not be aligned with real human preferences.
- Evaluation guidelines can be hard to write.
- Generating labels from commercial language models may violate their terms of service.



PROBLEM 4

Knowledge gaps and hallucination

Improving language models by retrieving from trillions of tokens

Sebastian Borgeaud[†], Arthur Mensch[†], Jordan Hoffmann[†], Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae[‡], Erich Elsen[‡] and Laurent Sifre^{†,‡}
All authors from DeepMind, [†]Equal contributions, [‡]Equal senior authorship

We enhance auto-regressive language models by conditioning on document chunks retrieved from a large corpus, based on local similarity with preceding tokens. With a 2 trillion token database, our Retrieval-Enhanced Transformer (RETRO) obtains comparable performance to GPT-3 and Jurassic-1 on the Pile, despite using 25× fewer parameters. After fine-tuning, RETRO performance translates to downstream knowledge-intensive tasks such as question answering. RETRO combines a frozen BERT retriever, a differentiable encoder and a chunked cross-attention mechanism to predict tokens based on an order of magnitude more data than what is typically consumed during training. We typically train RETRO from scratch, yet can also rapidly RETROfit pre-trained transformers with retrieval and still achieve good performance. Our work opens up new avenues for improving language models through explicit memory at unprecedented scale.

1. Introduction

Language modelling (LM) is an unsupervised task that consists of modelling the probability of text, usually by factorising it into conditional next-token predictions $p(x_1, \dots, x_n) = \prod_i p(x_i | x_{<i})$. Neural networks have proven to be powerful language models, first in the form of recurrent architectures (Graves, 2013; Jozefowicz et al., 2016; Mikolov et al., 2010) and more recently in the form of Transformers (Vaswani et al., 2017), that use attention to contextualise the past. Large performance improvements have come from increasing the amount of data, training compute, or model parameters. Transformers have been scaled from 100 million parameter models in seminal work to over hundred billion parameters (Brown et al., 2020; Radford et al., 2019) in the last two years which has led to models that do very well on a wide array of tasks in a zero or few-shot formulation. Increasing model size predictably improves performance on a wide range of downstream tasks (Kaplan et al., 2020). The benefits of increasing the number of parameters come from two factors: additional computations at training and inference time, and increased memorization of the training data.

RETRO - Embedding Search

Look up sentence in token database and get the most is similar completion sentence.

It's a great way to add private data to models in a scalable way. It also is privacy preserving, unlike finetuning.

WebGPT: Browser-assisted question-answering with human feedback

Reiichiro Nakano* Jacob Hilton* Suchir Balaji* Jeff Wu Long Ouyang
Christina Kim Christopher Hesse Shantanu Jain Vineet Kosaraju
William Saunders Xu Jiang Karl Cobbe Tyna Eloundou Gretchen Krueger
Kevin Button Matthew Knight Benjamin Chess John Schulman

OpenAI

Abstract

We fine-tune GPT-3 to answer long-form questions using a text-based web-browsing environment, which allows the model to search and navigate the web. By setting up the task so that it can be performed by humans, we are able to train models on the task using imitation learning, and then optimize answer quality with human feedback. To make human evaluation of factual accuracy easier, models must collect references while browsing in support of their answers. We train and evaluate our models on ELI5, a dataset of questions asked by Reddit users. Our best model is obtained by fine-tuning GPT-3 using behavior cloning, and then performing rejection sampling against a reward model trained to predict human preferences. This model's answers are preferred by humans 56% of the time to those of our human demonstrators, and 69% of the time to the highest-voted answer from Reddit.

1 Introduction

A rising challenge in NLP is long-form question-answering (LFQA), in which a paragraph-length answer is generated in response to an open-ended question. LFQA systems have the potential to become one of the main ways people learn about the world, but currently lag behind human performance [Krishna et al., 2021]. Existing work tends to focus on two core components of the task, information retrieval and synthesis.

In this work we leverage existing solutions to these components: we outsource document retrieval to the Microsoft Bing Web Search API,² and utilize unsupervised pre-training to achieve high-quality synthesis by fine-tuning GPT-3 [Brown et al., 2020]. Instead of trying to improve these ingredients, we focus on combining them using more faithful training objectives. Following Stiennon et al. [2020], we use human feedback to directly optimize answer quality, allowing us to achieve performance competitive with humans.

We make two key contributions:

WebGTP - Web Search

Why not give LLMs access web search? Search results can be added to the prompt context to ground responses.

This is quite easy! But it requires **prompt chaining**.

Retrieval-based context augmentation

PROS

- Very easy to extend LLMs to use tools like web search or embedding search.
- Retrieval use is natural path to tie into propriety systems and data.
- Retrieval is privacy preserving. No customer data is in your training set!

CONS

- Models don't learn to reason fully over the external data
- You are still stuck with the context limit of models (~4k tokens)
- Embeddings of queries and documents don't always match (but there are fixes...)



PROBLEM 5

Taking action

Toolformer: Language Models Can Teach Themselves to Use Tools

Timo Schick Jane Dwivedi-Yu Roberto Dessì† Roberta Raileanu
Maria Lomeli Luke Zettlemoyer Nicola Cancedda Thomas Scialom

Meta AI Research †Universitat Pompeu Fabra

Abstract

Language models (LMs) exhibit remarkable abilities to solve new tasks from just a few examples or textual instructions, especially at scale. They also, paradoxically, struggle with basic functionality, such as arithmetic or factual lookup, where much simpler and smaller models excel. In this paper, we show that LMs can teach themselves to *use external tools* via simple APIs and achieve the best of both worlds. We introduce *Toolformer*, a model trained to decide which APIs to call, when to call them, what arguments to pass, and how to best incorporate the results into future token prediction. This is done in a self-supervised way, requiring nothing more than a handful of demonstrations for each API. We incorporate a range of tools, including a calculator, a Q&A system, a search engine, a translation system, and a calendar. Toolformer achieves substantially improved zero-shot performance across a variety of downstream tasks, often competitive with much larger models, without sacrificing its core language modeling abilities.

Introduction

The New England Journal of Medicine is a registered trademark of [QA("Who is the publisher of The New England Journal of Medicine?") → Massachusetts Medical Society] the MMS.

Out of 1400 participants, 400 (or [Calculator(400 / 1400) → 0.29] 29%) passed the test.

The name derives from "la tortuga", the Spanish word for [MT("tortuga") → turtle] turtle.

The Brown Act is California's law [WikiSearch("Brown Act") → The Ralph M. Brown Act is an act of the California State Legislature that guarantees the public's right to attend and participate in meetings of local legislative bodies.] that requires legislative bodies, like city councils, to hold their meetings open to the public.

Figure 1: Exemplary predictions of Toolformer. The model autonomously decides to call different APIs (from top to bottom: a question answering system, a calculator, a machine translation system, and a Wikipedia search engine) to obtain information that is useful for completing a piece of text.

Toolformer

WebGPT and RETRO are fixed chains... Can we broaden this to allow LLMs to decide what tools to use?

Yes! It just requires smart prompting.

Toolformer and ReAct are two easy approaches.

Adding skills and tool use

PROS

- Very easy to extend LLMs to use external tools like APIs and native functions with clever prompting.
- Tool use allows LLMs to act in the real world and automate external systems
- Tool use is privacy preserving. No customer data is in your training set!
- Tool use and prompt chaining enables complex task completion via smaller, simpler tasks

CONS

- Tool use and prompt chaining are more complicated than one API call.
- Chained models sometimes go off track. You'll likely need retries, etc.
- Chaining together models and tools can be slow if you're not careful.
- Giving LLMs tools opens the door to safety concerns!

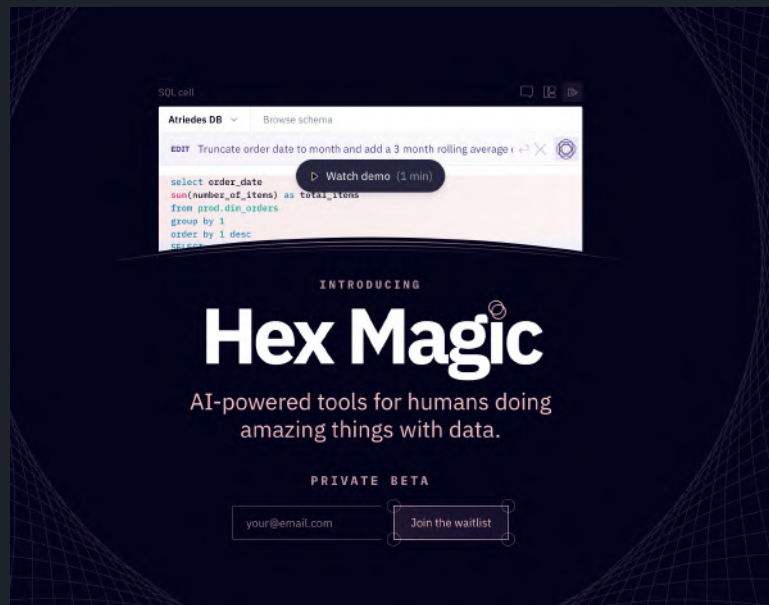


Let's recap...



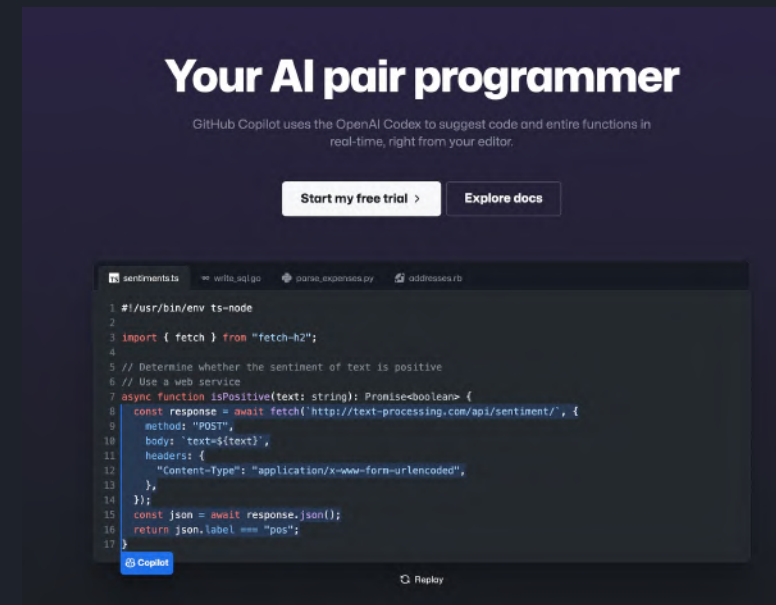
What can we do with all this power?

EMERGING CLASSES OF GENERATIVE AI PRODUCTS



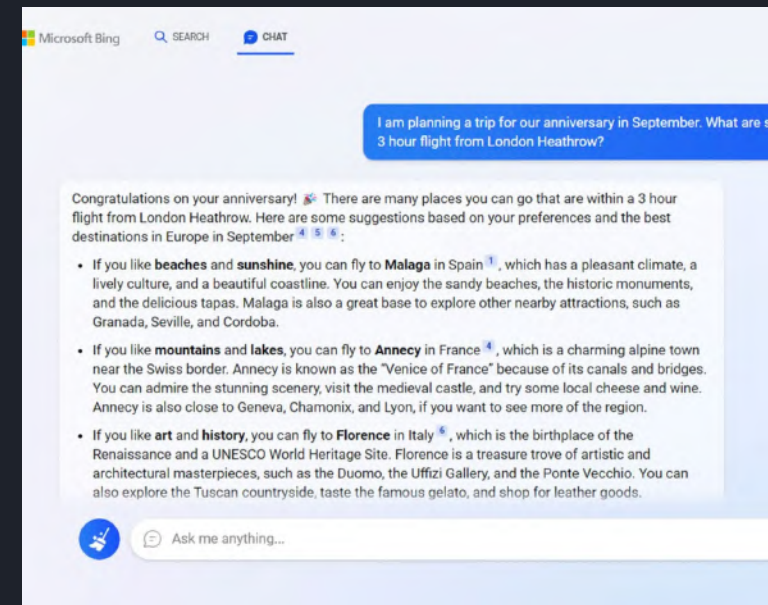
AI features

Lots of nice-to-have AI features within existing products.



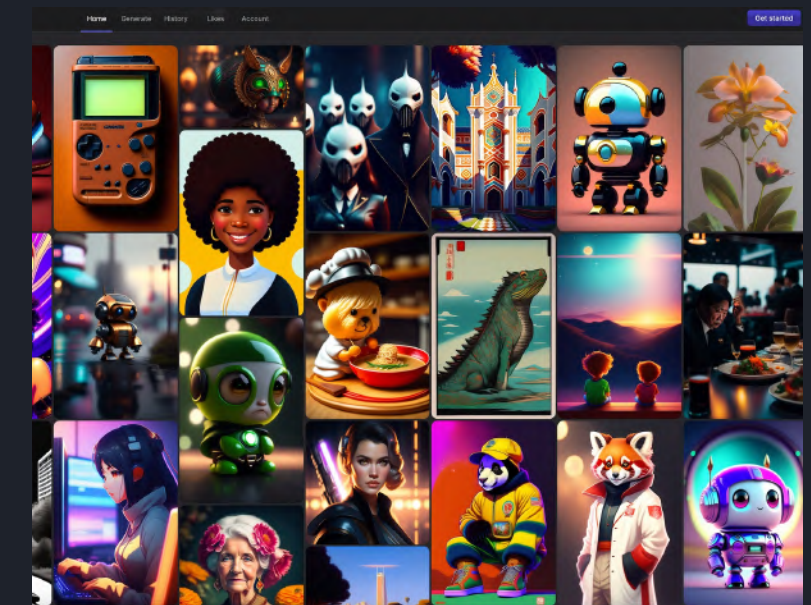
Copilots

Productivity sidekicks that work with new or existing tools.



Category disrupters

Fundamentally new workflows for existing categories.



Category creators

Brand new categories of products, enabled by AI.

A laptop is shown from a low angle, open and glowing. The screen and keyboard area are illuminated with a vibrant, multi-colored light that transitions from a bright cyan on the left to a warm orange and red on the right. The laptop is positioned on a dark surface, and the background is a deep, dark blue/black. The overall mood is futuristic and inspiring.

Let's build an awesome future!



Thank you!