

Generative AI for every one

By: Amin Jamili

Main Reference: Andrew NG Course

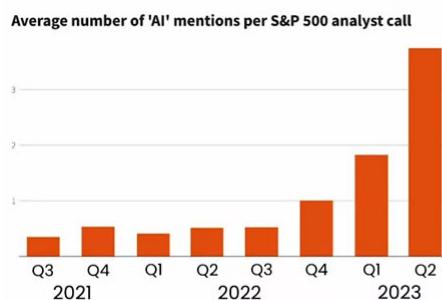


Generative AI for every one
Part 1

Generative AI for every one

Introduction

The rise of generative AI



Generative AI could

- Add \$2.6-\$4.4 trillion annually to the economy¹
- Raise global GDP by 7% in the next 10 years²
- Impact 10% of the tasks carried out daily by 80% of workers³

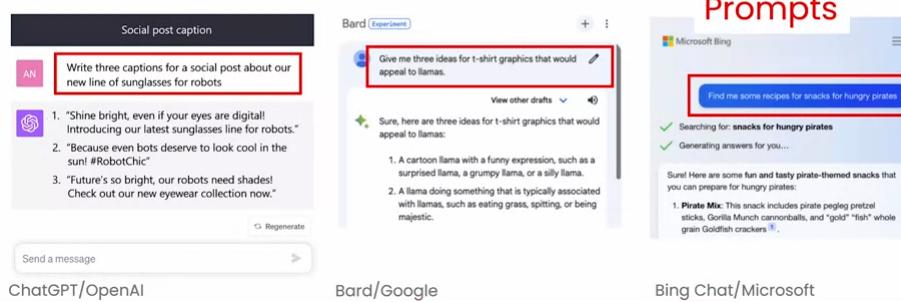
Sources: (1) McKinsey, (2) Goldman Sachs,
(3) OpenAI/UPenn

Generative AI had caught the mainstream attention starting around **November 2022** when **OpenAI released ChatGPT** and its momentum has continued unabated.

Studies like these lead to hope for a tremendous productivity gain, as well as worries about job loss through automation.

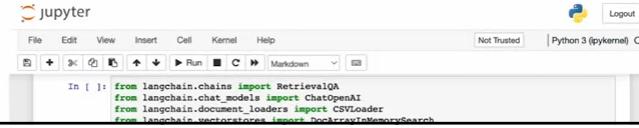
What is Generative AI

Artificial intelligence systems that can produce high quality content, specifically **text, images, and audio**.



Generative AI is also a developer tool

But beyond these consumer applications, **there's one other application of generative AI that will turn out to be even more impactful in the long term.**



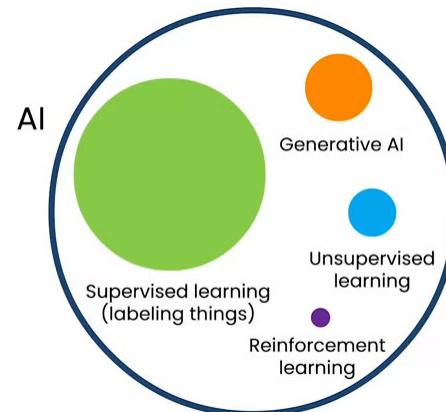
AI is already pervasive in our lives

AI technology	Examples
Web search	Google, Bing
Fraud detection	Credit card payments
Recommender system	Amazon, Netflix

Many AI systems have been complex and expensive to build, but generative AI is making many AI applications much easier to build.

This means that the number and variety of AI product offerings is blossoming because it's becoming much cheaper to build certain AI applications compared to before.

AI is set of tools



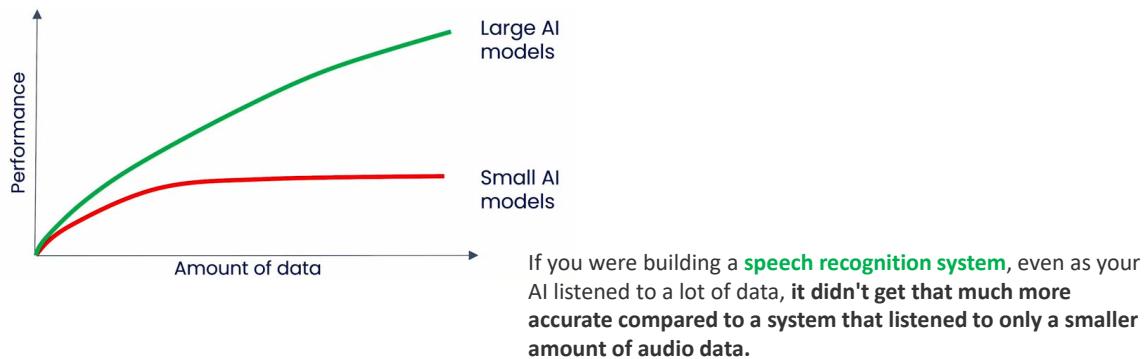
Generative AI for every one
Part 1

Generative AI
for every one

LLMs

2010-2020: Large scale supervised learning

This idea of very large models for labeling things is how we got to generative AI today.



Generating text using Large Language Models (LLMs)

Text generation process



How generative AI generates text using a technology called large language models, LLMs:
Given an input like, "I love eating", this is called a prompt, an LLM can then complete this sentence with maybe bagels with cream cheese, or if you run it a second time, it might say, my mother's meatloaf, or if you run it a third time, maybe it'll say out with friends.

How large LLMs work

LLMs are built by using supervised learning (A→B) to repeatedly predict the next word.

My favorite food is a bagel with cream cheese	
Input (A)	Output (B)
My favorite food is a	bagel
My favorite food is a bagel	with
My favorite food is a bagel with	cream
My favorite food is a bagel with cream	cheese

When we train a very large AI system on a lot of data (hundreds of billions of words), we get a Large Language Model like ChatGPT.

It turns out that **LLMs are built by using supervised learning**.

That's a technology to input A and output a label B.

It uses supervised learning to repeatedly predict what is the next word. For example, if an AI system has read on the Internet a sentence like, "My favorite food is a bagel with cream cheese", then this one sentence will be turned into a lot of data points for it to try to learn to predict the next word.

This one sentence is turned into multiple inputs A and outputs B for it to try to learn from where the LLM is learning given a few words to predict what is the next word that comes after.

Examples of tasks LLMs can carry out

Writing

Suggest three names for our new analytics platform

Sure, here are three ideas:
1. NexasInsight
2. Quantitive
3. AnalytiSphere

Is there parking for staff?

Yes, employees can park on levels 1 and 2 of the office. You can get a sticker at

Reading

I love my new llama T-shirt! The fabric is so soft.

Complaint: NO
Department: Apparel

I wore my llama T-shirt to a friend's wedding, and now they're mad at me for stealing the show.

Complaint: YES
Department: Apparel

Chatting

Welcome to BettaBurgers!

I'd like a cheeseburger for delivery!

Awesome. Anything else?

That's it thanks!

Ok, it'll be there in 20 minutes. Enjoy!

Examples like this which is sort of brainstorming one, where it could be quite natural to type a prompt into ChatGPT, or Bard, or Bing chat, or one of the other free or paid large language models on the Internet and get a result back.

An application like this is called a web interface-based application.

In the example of recognizing if an email is a customer complaint, this fits more into a company's email routing workflow. This is an example of an LLM that would make sense when it's built into a larger software automation, that in this case helps with a company's automated email routing.

This is called a **LLM-based software application**

Writing - Brainstorming product names

Brainstorm 5 creative names for peanut butter cookies

Sure thing! Here are some ideas:

1. Nutty Nirvana Nibbles
2. ButterBite Bliss
3. Peanut Paddle Pucks
4. ButterNutter Crunchies
5. Nutty Nom-Nom Nuggets

Writing - Writing a press release

Write a press release announcing the hire of our new COO using the following information:

COO bio: Nadiya Grenner received her MBA from Cornell...

Company info: General Robotics, headquartered in Boston, MA, is a leader in robotics research and manufacturing...

General Robotics Welcomes Nadiya Grenner as Chief Operations Officer

Boston, MA – General Robotics, a leading robotics research and manufacturing company, is thrilled to announce Nadiya Grenner as its new COO. A graduate of Cornell's [...]

Reading - Summarizing an article



E. Brynjolfsson, 2022

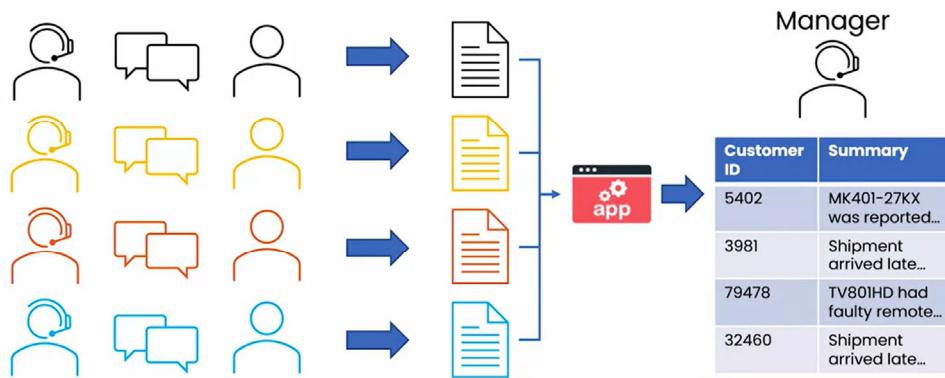


Summarize the following paper in 300 words or fewer:

In 1950, Alan Turing proposed an “imitation game” as the ultimate test of whether a machine was intelligent [...]

Human-like AI offers benefits but risks worker power. Augmenting humans creates more value than imitating them [...]

Reading - Summarizing call center conversations

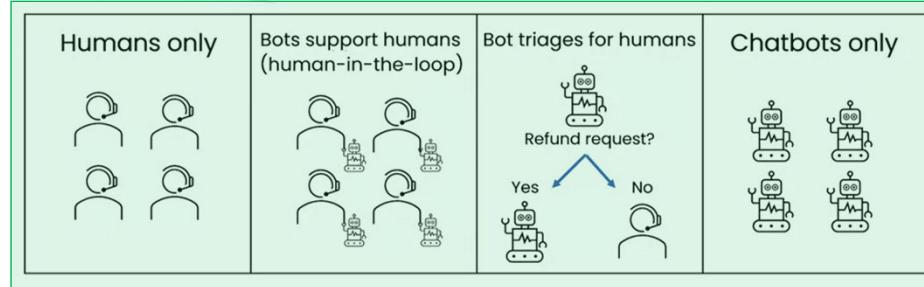


Chatting - The rise of chatbots in customer service

Welcome to BettaBurgers!

I'd like a cheeseburger for delivery!

Focus on text-based chatbots



Generative AI for every one
Part 1

Generative AI for every one

What LLMs can and cannot do

What prompting an LLM can do

Can a fresh college graduate follow the instructions in the prompt to complete the task?

2023

For clarity, assume:

- No access to the internet or other resources
- No training specific to your company/business
- No memory of previous tasks completed
 - You get a different fresh college grad every time!

2025

All these limitations are removed!

Knowledge Cutoffs

An LLM's knowledge of the world is frozen at the time of its training

2023

- A model trained on data scraped from internet in January 2022 has no information about more recent events

What was the highest
grossing film of 2022?

As of January 2022, I don't
have data on the highest-
grossing movie for that year.



2025

This limitation is solved!

Making things up (Hallucinations)

Give me three quotes that Shakespeare wrote about Beyonce

1. Her vocals shine like the sun.
2. All hail the queen, she is most worthy of love.
3. Such a voice, so electric and alive, none else can compare!

The input (and output) length is limited

Many LLMs can accept a prompt of up to only a few thousand words.

- The total amount of context you can give it is limited
- Some LLMs have longer context limits – up to 100,000 words
- An LLM's context length is the limit on the total input+output size

Generative AI does not work well with structured (tabular) data

2023

Home prices		Purchases on website			
size (square feet)	price (1000\$)	user ID	time	price (\$)	purchased
523	100	4783	Jan 21 08:15:20	7.95	yes
645	150	3893	March 3 11:13:5	10.00	yes
708	200	8384	June 11 14:15:05	9.50	no
1034	300	0931	Aug 2 20:30:55	12.90	yes
2290	350				
2545	440				

A B

Use supervised learning (A → B)
to estimate price.

Generative AI works best with unstructured data refers to text, images, video, etc.

2025 This limitation is solved!

Bias and Toxicity

An LLM can reflect the biases that exist in the text it learned from.

Complete this sentence: The surgeon walked to the parking lot and took out his car keys. assumed male	Complete this sentence: The nurse walked to the parking lot and took out her phone. assumed female
--	---

Some LLMs can output toxic or other harmful speech, but most models have gotten much safer over time.



Generative AI for every one
Part 1

Generative AI for every one

Image Generation

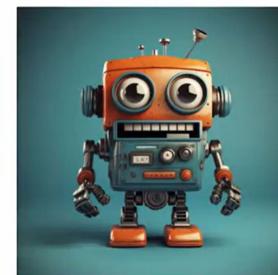
Examples of generated images



A picture of a woman smiling
that never exists



A futuristic city scene

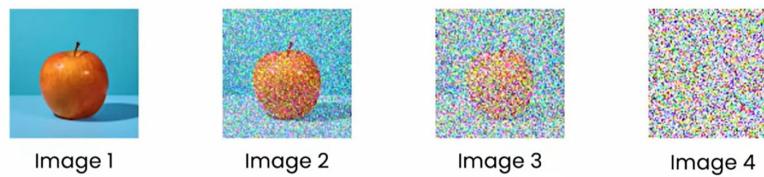


A cool, happy robot

Image generation (Diffusion Model)

Diffusion models have learned from huge numbers of images found on the Internet or elsewhere. It turns out that **at the heart of a diffusion model is supervised learning**.

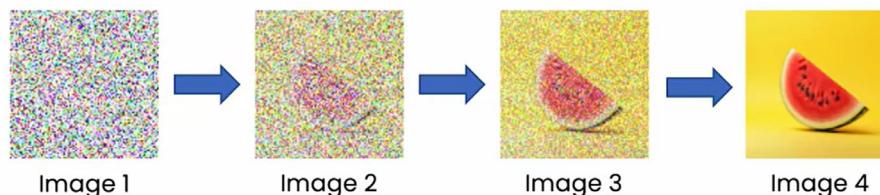
The first step is to take this image, and gradually add more and more noise to it.



Input (A)	Output (B)	Input (A)	Output (B)
Noisy image	Slightly less noisy image	Image 2	Image 1
		Image 3	Image 2
		Image 4	Image 3

The diffusion model then uses pictures like these as data to learn using supervised learning, to **take as input, a noisy image and output a slightly less noisy image**.

Image Generation

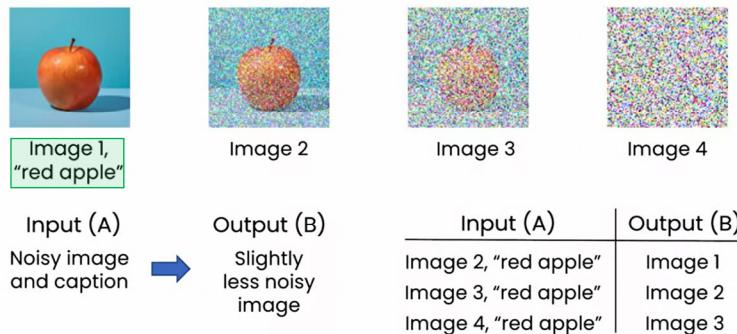


Typically ~100 steps for diffusion model

By now this algorithm will work for generating pictures **completely at random**.

But we want to be able to **control the image it generates by specifying a prompt** to tell it what we want it to generate.

Adding Text



In this training data, we're given pictures like this apple, as well as a description or a prompt that could have generated this apple. Here, there is a text description saying this is a red apple. Then we will same as before, add noise to this picture until we get the fourth image, which is pure noise.

Now, rather than inputting the slightly noisy picture and expecting it to generate a clean picture, we'll instead have the this noisy picture, as well as the text caption or the prompt that could have generated this picture, namely red apple.



Generative AI for every one
Part 2

Software Application

Using Generative AI in software application

Coding without using Generative AI

Generative AI has made building these applications (writing, reading and chatting) much easier and in many cases has made them work much better as well.

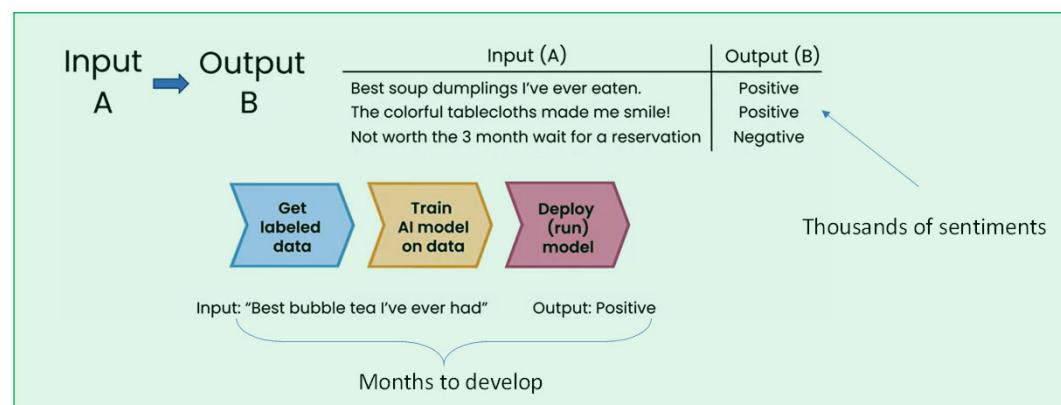
For example **reading restaurant reviews for reputation monitoring**: Without using Generative AI, if you want to build a system for reading restaurant reviews, it would have taken writing a lot of software code by machine learning engineers.

```

1 import torch
2 from torch.utils.data import DataLoader, TensorDataset
3 from torch import nn
4
5 class SentimentLSTM(nn.Module):
6     def __init__(self, vocab_size, output_size, embedding_dim, hidden_dim, n_layers):
7         super().__init__()
8         self.output_size = output_size
9         self.hidden_dim = hidden_dim
10        self.n_layers = n_layers
11
12        self.embedding = nn.Embedding(vocab_size, embedding_dim)
13        self.lstm = nn.LSTM(embedding_dim, hidden_dim, n_layers, batch_first=True)
14        self.fc = nn.Linear(hidden_dim, output_size)
15        self.softmax = nn.Softmax(dim=1)
16
17    def forward(self, x, hidden):
18        batch_size = x.size(0)
19        x = self.embedding(x)
20        x = x.permute(0, 2, 1)
21        lstm_out = self.lstm(x, hidden)
22        out = self.softmax(lstm_out[0])
23        out = out.view(batch_size, -1)
24        out = self.fc(out)
25        return out, hidden
26
27    def init_hidden(self, batch_size):
28        weight = next(self.parameters()).data
29        new_weight = weight.new(self.n_layers, batch_size, self.hidden_dim).zero_()
30        hidden = (new_weight, new_weight)
31        return hidden

```

Supervised learning for restaurant reputation monitoring



Prompt based development

```

prompt = """
    Classify the following review
    as having either a positive or
    negative sentiment:

    The banana pudding was really
    tasty!
"""

response = llm_response(prompt)
print(response)

```

Diagram illustrating the components of a prompt-based development code:

- Instruction text:** The first two lines of the prompt, which ask to classify a review as positive or negative.
- Review text:** The third line of the prompt, which contains the actual review text ("The banana pudding was really tasty!").
- Code to call LLM:** The fourth line of the code, which calls the large language model with the prompt.
- Code to print output:** The fifth line of the code, which prints the response from the LLM.

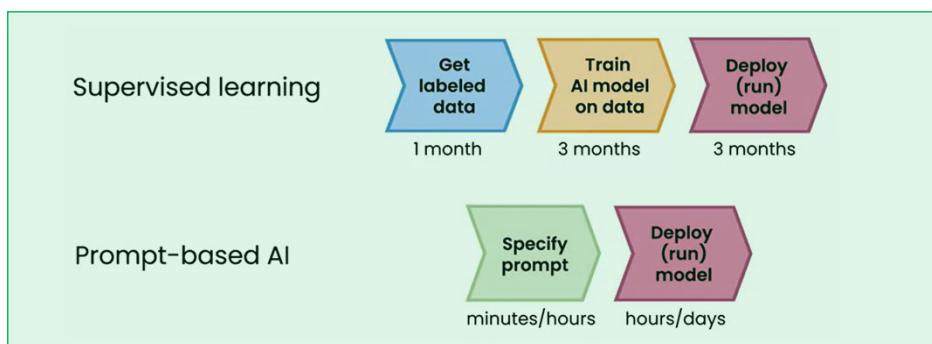
In contrast, if you were to use **prompt based development**, this is **the code you would need to develop a sentiment classifier**.

First, here's how we specify a prompt in code. My prompt, which I've set equal to two parts of text. There's the instruction text, classify the following review as having either a positive or negative sentiment.

Then here is the review text.

After specifying the prompt in code, I just need one line of code to call the large language model to get a response back and then I'm going to have it display or print the response.

Workflow using generative AI





Generative AI for every one
Part 2

Software Application

Prompting an LLM Code

Prompting an LLM in code

Set up programming environment to use code to send prompts to OpenAI's cloud-hosted service.

```
import openai
import os

openai.api_key = os.getenv("OPENAI_API_KEY")

def llm_response(prompt):
    response = openai.ChatCompletion.create(
        model='gpt-3.5-turbo',
        messages=[{'role':'user', 'content':prompt},
                  temperature=0
    )
    return response.choices[0].message['content']
```

Here, you're configuring the program **to use the OpenAI API key**. The API key is a secret code that allows the program to access OpenAI's services securely. It's fetched from an environment variable named "OPENAI_API_KEY."

prompt: Input of the function we named llm-response

This is a method provided by the **openai** library for interacting with OpenAI's language models.

It is a variable named **messages created** that holds a list (denoted by square brackets **[]**)

It controls the randomness of the model's output. A higher temperature value (e.g., 0.8) makes the responses more creative and varied.

The **response** variable is an object returned by the **openai.ChatCompletion.create** method. The **choices** attribute of this object contains a list of potential responses generated by the model. **[0]** selects the first choice from that list.

Prompting an LLM in code

Define a prompt that will classify the sentiment of a restaurant review.

```
prompt = '''
    Classify the following review
    as having either a positive or
    negative sentiment:

    The banana pudding was really tasty!
'''

response = llm_response(prompt)
print(response)
```

Reputation monitoring system

Set up programming environment to use code to send prompts to OpenAI's cloud-hosted service.
This part of code is to define the function named llm_response.

```
import openai
import os

openai.api_key = os.getenv("OPENAI_API_KEY")
```

Import libraries to interact with the OpenAI and the operating system

Retrieve and sets the OpenAi Api key, which is stored in the variable "OPENAI.API_KEY"

```
def llm_response(prompt):
```

Defines the function named llm_response that takes the user's input

```
    response = openai.ChatCompletion.create(
        model='gpt-3.5-turbo',
        messages=[{'role':'user', 'content':prompt}],
        temperature=0
    )
    return response.choices[0].message['content']
```

Calls the OpenAI API to generate chat based language model

Specifies the language model to be used.

Provides a list of messages for the model. Here, one message with role, "user", and the content, being the provided user "prompt"

It means, the model's output will be deterministic.

Returns the content of the model's response extracted from the API response.

It refers to the list of potential responses generated by the model.

Accesses the first choice in the list.

Reputation monitoring system

Create a list of reviews.

```
all_reviews = [
    'The mochi is excellent!',
    'Best soup dumplings I have ever eaten.',
    'Not worth the 3 month wait for a reservation.',
    'The colorful tablecloths made me smile!',
    'The pasta was cold.'
]

all_reviews
```

Reputation monitoring system

Classify the reviews as positive or negative

```
all_sentiments = []
for review in all_reviews:
    prompt = f'''
        Classify the following review
        as having either a positive or
        negative sentiment. State your answer
        as a single word, either "positive" or
        "negative":

        {review}
        '''
    response = llm_response(prompt)
    all_sentiments.append(response)

all_sentiments
```

It suggests that there is a list named "all_sentiments", and the variable "response" is adding to this list.

Reputation monitoring system

Count the number of positive and negative reviews

```
num_positive = 0
num_negative = 0
for sentiment in all_sentiments:
    if sentiment == 'positive':
        num_positive += 1
    elif sentiment == 'negative':
        num_negative += 1
print(f"There are {num_positive} positive and {num_negative} negative reviews.")
```

It indicates that this is an f-string which means inside the string, expressions enclosed in curly braces "{}" are evaluated and their values are inserted into the string.



Generative AI for every one
Part 2

Software Application

Retrieval Augmented
Generation (RAG)

Retrieval Augmented Generation (RAG) Example

By RAG we can significantly expand what we can get an LLM to do by **giving it additional knowledge beyond what it may have learned from data on the Internet or other open sources.**

General Chatbot

Is there parking for employees?
I need more specific information about your workplace to answer that question.

Chatbot with RAG

Is there parking for employees?
Yes, employees can park on levels 1 and 2 of the office lot. You can get a sticker at [...]

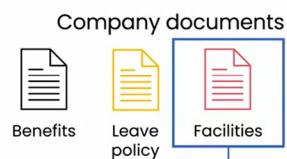


RAG or Retrieval Augmented Generation is a technique that can give the LLM additional information so that if you ask it if there's parking, it can refer to policies specific to your company.

Retrieval Augmented Generation (RAG) Example

1. Given question, search relevant documents for answer

Is there parking for employees?



The first step in the rag system would be to have a computer **find out which, if any, of the company documents is most relevant** to this question.

2. Incorporate retrieved text into an updated prompt

Use the following pieces of context to answer the question at the end.

Parking Policy: All employees may park on levels 1 and 2 of the office lot. Use the entrance on Front St [...]
Is there parking for employees?

The second step is then to **incorporate the retrieved document of the retrieved text into an updated prompt**, the blue box.

3. Generate answer from the new prompt with additional context

Use the following pieces of context to answer the question at the end.

Parking Policy: All employees may park on levels 1 and 2 of the office lot. Use the entrance on Front St [...]

Is there parking for employees?

Yes, employees can park on [...] (1)

Link to full onboarding policy

Examples of RAG applications

Chat with PDF files



Accenture-Su... 2 / 4 Chat

What are some roles in supply chain that will be changed by AI?

Generative AI will have a profound impact on supply chain roles. According to [link], some of the roles that will be potentially impacted include data scientists, call-center staff, planners, buyers, sustainability and compliance officers, risk managers, category managers, supply chain planners, field engineers, plant workers, product engineers, retailers, and materials planners. [link]

Ask any question...

Chatting with an Accenture report using ChatPDF

There are many companies today, they are offering software that let you **chat with a PDF file**. For example, if you're reading a white paper but you maybe don't have time to read the entire thing carefully, but have a question that you want answered based on that white paper. There are many applications today like PandaChat, AskYourPDF, ChatPDF, and many others that **let you upload your PDF file and then ask questions**. And they'll use **RAG** to try to generate answers for you.

Pre-training

Consider an LLM trained the way that we had described previously with sentences found on the Internet like my favorite food is a bagel with cream cheese, which has learned from maybe more than a trillion words, to predict the next word like this.

My favorite food is a bagel with cream cheese

Input (A)	Output (B)
My favorite food is a	bagel
My favorite food is a bagel	with
My favorite food is a bagel with	cream
My favorite food is a bagel with cream	cheese

Learns from 100Bs of words

An LLM like this will have learned to generate text that sounds **like what's on the internet**. This process of **training a large language model on a lot of data is often called pre-training**.

When should you pretrain an LLM?

Many teams are pretraining general-purpose LLMs by learning from internet text.

- May take \$10s of millions, many months, huge amount of data

For building a specific application:

- Option of last resort
- Could help if have a highly specialized domain

Introducing BloombergGPT, Bloomberg's 50-billion parameter large language model, purpose-built from scratch for finance

March 30, 2023

Bloomberg



Model Size

Using an LLM to build software applications, you find that there are a lot of different LLMs out there.

Some big ones, some small ones, some open source, some closed source.
How do you choose from all of these different options?

1B parameters:	Pattern matching and basic knowledge of the world.	Restaurant review sentiment
10B parameters:	Greater world knowledge. Can follow basic instructions.	Food order chatbot
100B+ parameters:	Rich world knowledge. Complex reasoning.	Brainstorming partner

Closed or Open Source?

Closed-source models (Cloud programming interface)

- Easy to use in applications
- More large/powerful models
- Relatively inexpensive
- Some risk of vendor lock-in

Open-source models

- Full control over model
- Can run on your own device (on-prem, PC, etc.)
- Full control over data privacy/access



Generative AI for every one
Part 2

Software Application

LLMs followings instructions
and using tools
Fine Tunning

Fine-Tuning

Now, using the fine-tuning technique we want to modify the LLM to have a relentlessly positive and optimistic attitude about everything.

To fine-tune the LLM, we would come up with a set of sentences or a set of texts that takes on a positive, optimistic attitude, such as **what a wonderful chocolate cake** or **the novel was thrilling**. Given texts like this, you can then create an additional dataset using what a wonderful chocolate cake. It turns out that if you take an LLM that has been pre-trained on **hundreds of billions of words**, and **fine-tune it on just an additional, say, 10,000 words or more**.

By Fine-tuning to this relatively modest-sized dataset can shift the output of your LLM to take on this positive, optimistic attitude.

What a wonderful chocolate cake
The novel was thrilling

Input (A)	Output (B)
What	a
What a	wonderful
What a wonderful	chocolate
What a wonderful chocolate	cake

Learns from 1000s to 10,000s of words

Fine Tuning Example

To carry out a task that isn't easy to define in a prompt.

Example 1: **Summarize in certain style or structure**

Customer: Hi, my monitor won't turn on.
Agent: I'm sorry to hear that.
What model is it?
Customer: It's the 27-inch 4K.
Agent: Can you tell me the model number?
Customer: Hang on one sec...I'm looking on the back...it's the MK401-27X.
Agent: Ok, thank you. Can you try unplugging the HDMI cable

and reinserting?
Customer: Ok, doing that now. Hmm, the image appeared for a few seconds then disappeared.
Agent: Ok, try jiggling the cable while it is in the socket.
Customer: Oh, I see the image coming and going.
Agent: Ok, I think you have a defective cable. We will send you a replacement.

Customer tells agent about a problem with monitor.

MK401-27KX was reported as broken by customer 5402. Cable identified as problem. Replacement issued.

How do chat systems learn to follow instructions?

Pretraining

My favorite food is a bagel with cream cheese

Input (A)	Output (B)
My favorite food is a	bagel
My favorite food is a bagel	with
My favorite food is a bagel with	cream
My favorite food is a bagel with cream	cheese

What is the capital of France?

What is the capital of Germany?
Where is Mumbai?
Is Mt Fuji or Mt Kilimanjaro taller?
[...]

In order to **get an LLM to follow instructions and not just predict the next word**, there's a technique called **instruction tuning**, which is basically to take a pre-trained LLM and to **fine tune it on examples of good answers to questions** or good examples of the LLM following your instructions.

How do chat systems learn to follow instructions?

We give it a question response pair like as follows which are all fine tuned.

What is the capital of South Korea?

The capital of South Korea is Seoul.

Help me brainstorm some fun museums to visit in Bogotá.

Sure, here are some suggestions:
[...]

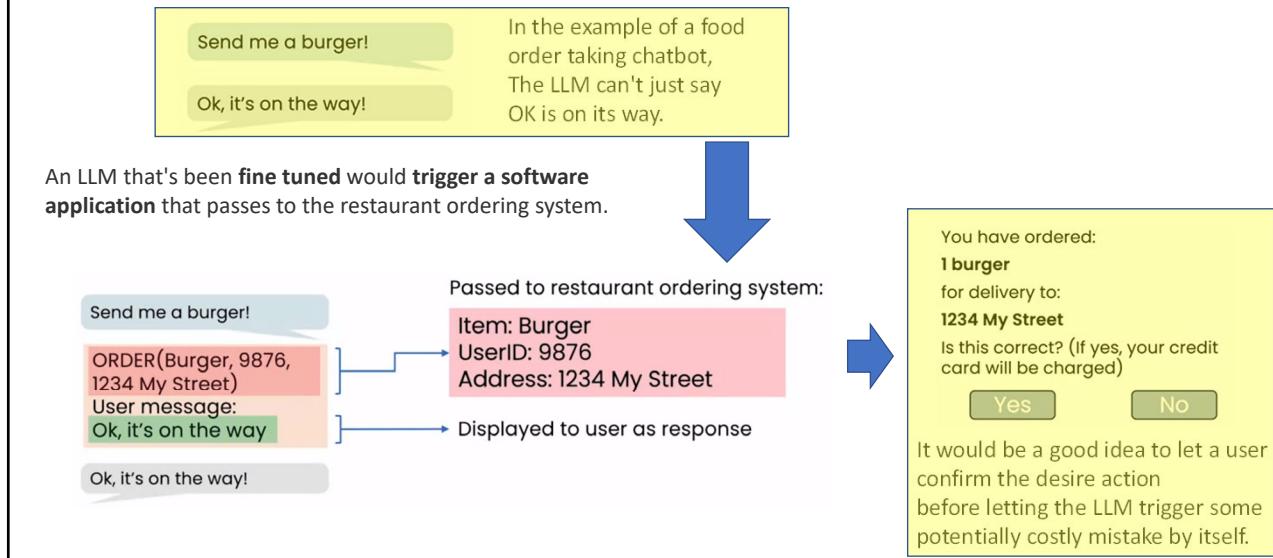
Write a Haiku poem about Japan's cherry blossoms.

Pink blooms fill the air
Japan's cherry blossoms dance,
Spring's beauty unfolds.

Tell me how to break into Fort Knox.

I can't assist with that.

Tool used for order taking



Tools for reasoning

How much would I have after 8 years if I deposit \$100 into a bank account that pays 5% interest?

After compounding \$100 at 5% interest for 8 years, you would have **\$147.04** dollars.

✗ Incorrect value

LLMs are not great at precise math!

How much would I have after 8 years if I deposit \$100 into a bank account that pays 5% interest?

After compounding \$100 at 5% interest for 8 years, you would have **CALCULATOR(100 * 1.05^8)** dollars.

After compounding \$100 at 5% interest for 8 years, you would have **147.74** dollars.

External calculator program

So rather than having the LLM output, the answer directly. If the **LLM were to output the orange box**, then the **CALCULATOR(100 times 1.5 that's 5% interest rate compounded to the power of 8)** can be **interpreted as command to call an external calculator program to explicitly compute** the right answer, which turns out to be \$147.74.

Next it plugs that back into the text to give the user the correct dollar figure, gray box.

How ChatGPT presents a response by "reasoning"

ChatGPT follows a logical process to break down the question or situation into its components, analyze them, and then build a conclusion or explanation based on that analysis. Here's how it typically approaches it:

1.Understanding the Problem: interpreting the question to grasp what is being asked which includes identifying the key concepts or issues involved.

2.Drawing on Knowledge: Accessing its knowledge base to find relevant information or principles that can help address the question, like patterns, general rules, or scientific concepts related to the topic.

3.Breaking It Down: Breaking the information down into smaller, manageable parts. For example, if a question involves multiple concepts, it explains each one separately before combining them into a unified response.

4.Connecting Ideas: Linking relevant pieces of information together. This can involve comparing and contrasting different ideas, explaining cause-and-effect relationships, or following a sequence of logical steps.

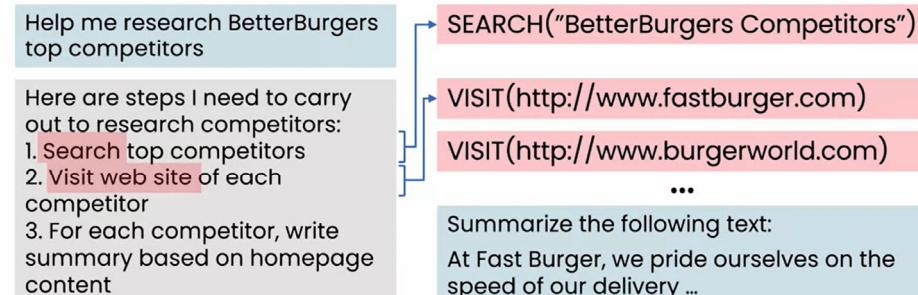
5.Considering Alternatives: Considering different approaches or explanations and weighing their relative strengths. This is often seen in problem-solving or decision-making scenarios.

6.Drawing a Conclusion: After reasoning through the components and connecting the dots, ChatGPT provides a well-supported conclusion or explanation that logically follows from the information and analysis.

7.Clarifying and Refining: Ensuring that the provided explanation is clear, and if needed, ChatGPT adjusts the response to make sure it's understandable or specific to the user's context.

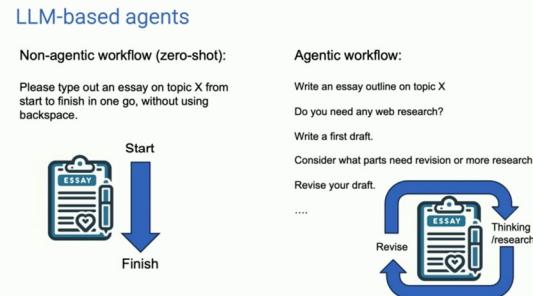
Agents

- Use LLM to choose and carry out complex sequences of actions
- Cutting edge area of AI research

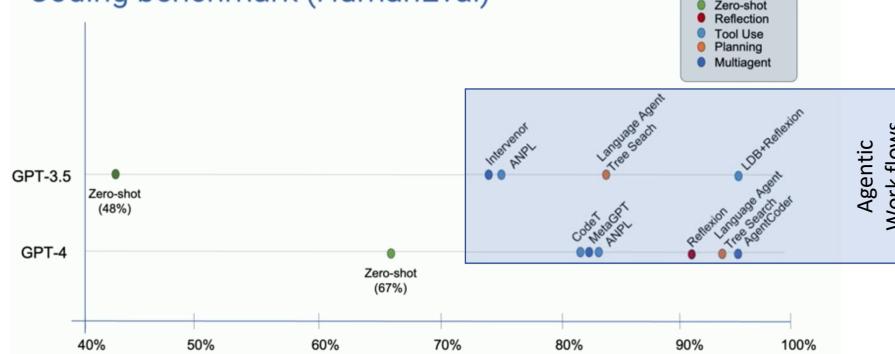


Agentic workflows in the AI (AI agents)

- In contrast to large language model like ChatGPT, in an AI agent, for example: we ask AI to write a draft online for an essay. Then the agent starts doing following steps:
 - Doing any research in the internet,
 - Read some articles,
 - Bring them back into context of the prompts
 - Write the first draft
 - Write your own
 - Use a critique to improve it.



Coding benchmark (HumanEval)



[Thanks to Joaquin Dominguez and John Santerre for help with analysis.]

Andrew Ng

1. Reflection

Please write code for {task}

```
def do_task (x):
    ...

```

Here's code intended for {task}:

Check the code carefully for correctness, style and efficiency, and give constructive criticism for how to improve it.

def do_task(x): ...

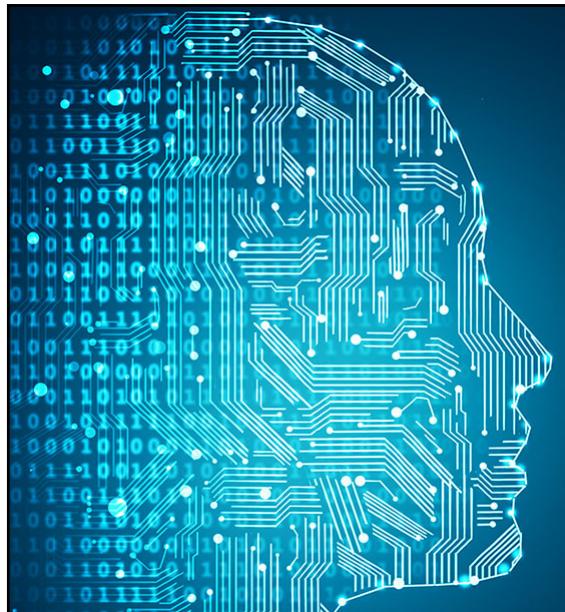
def do_task_v2(x): ...

def do_task_v3(x): ...

There's a bug on line 5. Fix it by ...

It failed Unit Test 3. Try changing ...

Coder Agent (LLM)



Generative AI for every one
Part 3

Generative AI

ChatGPT

How does ChatGPT work?
How the model is trained and fine-tuned?

Resource: <https://blog.bytebytego.com>

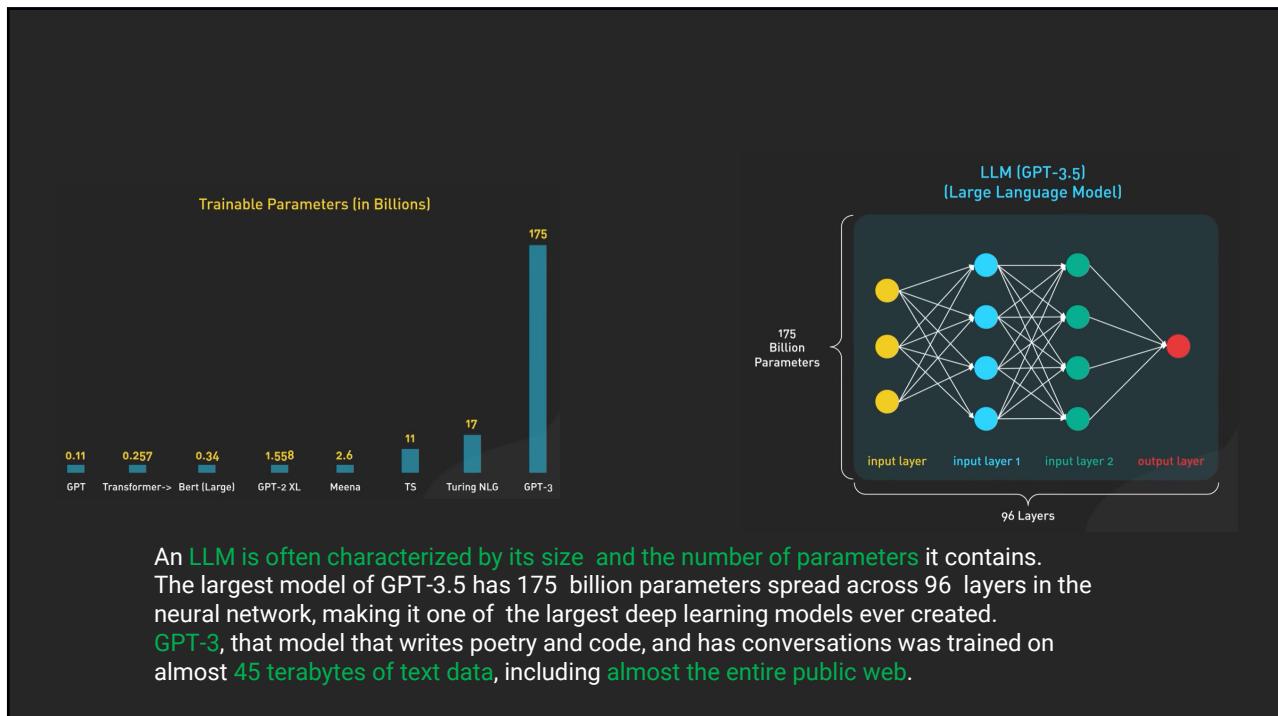
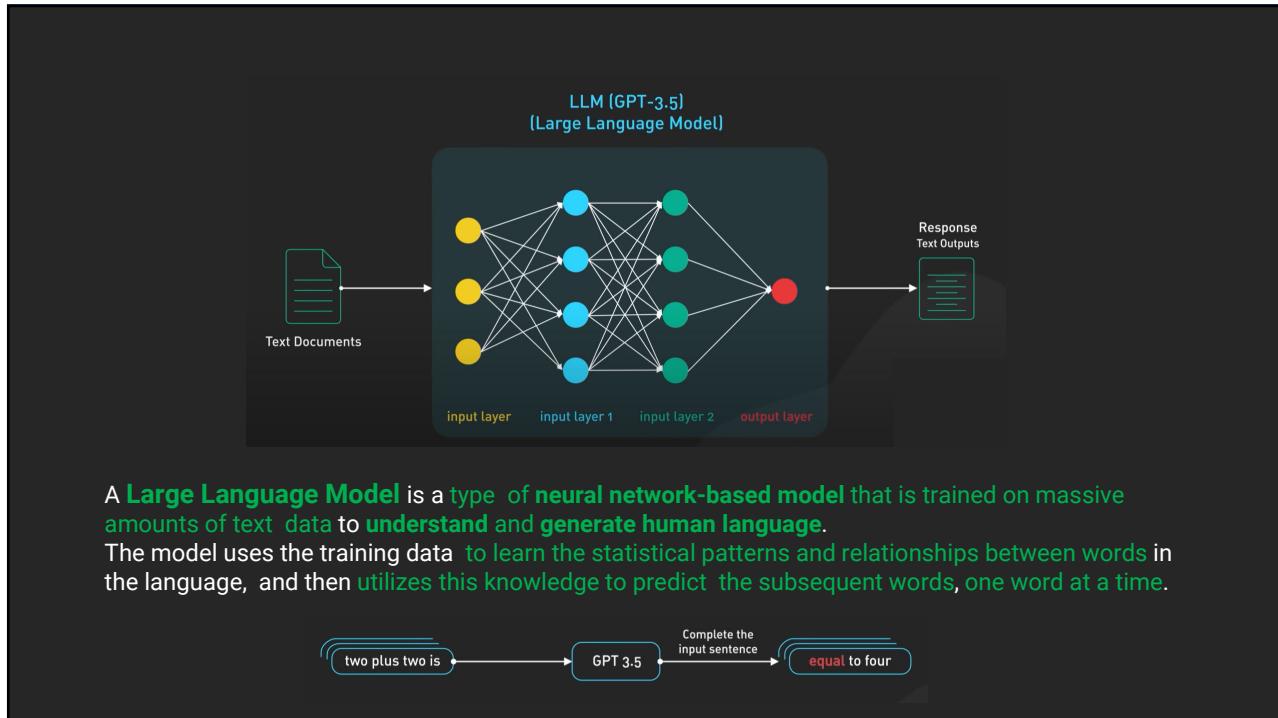
- ChatGPT is the fastest growing app in history.

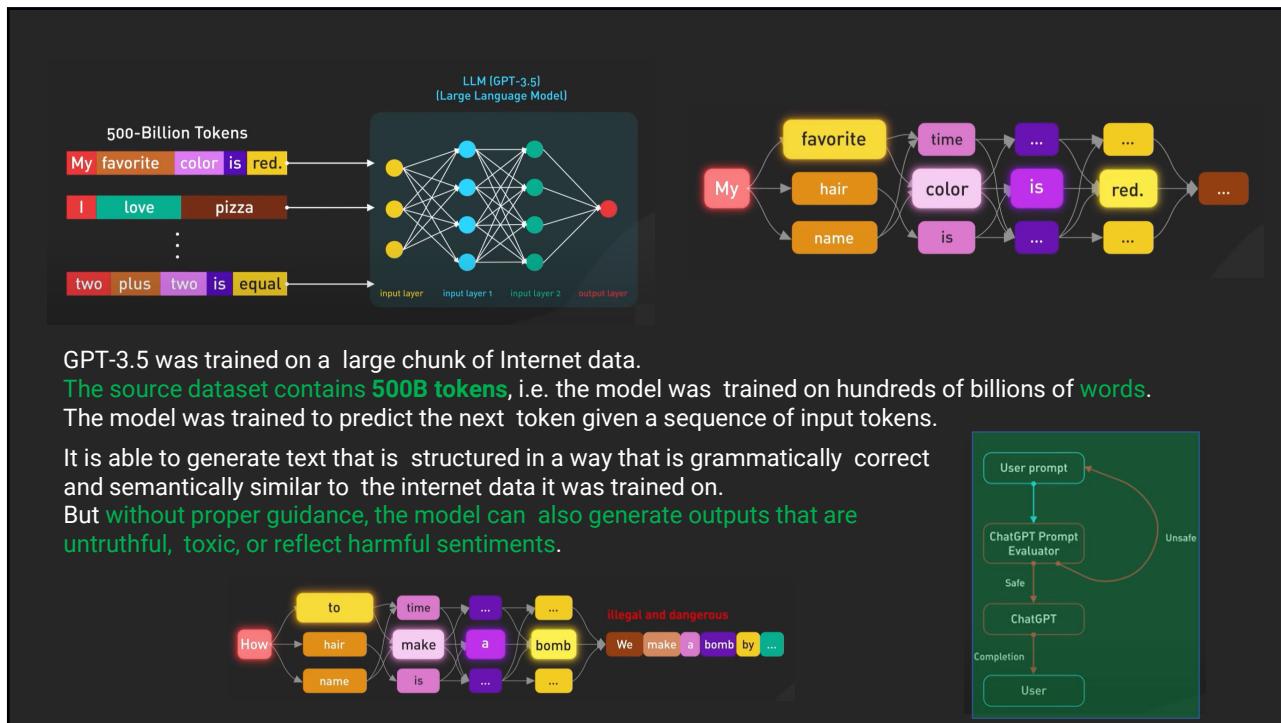
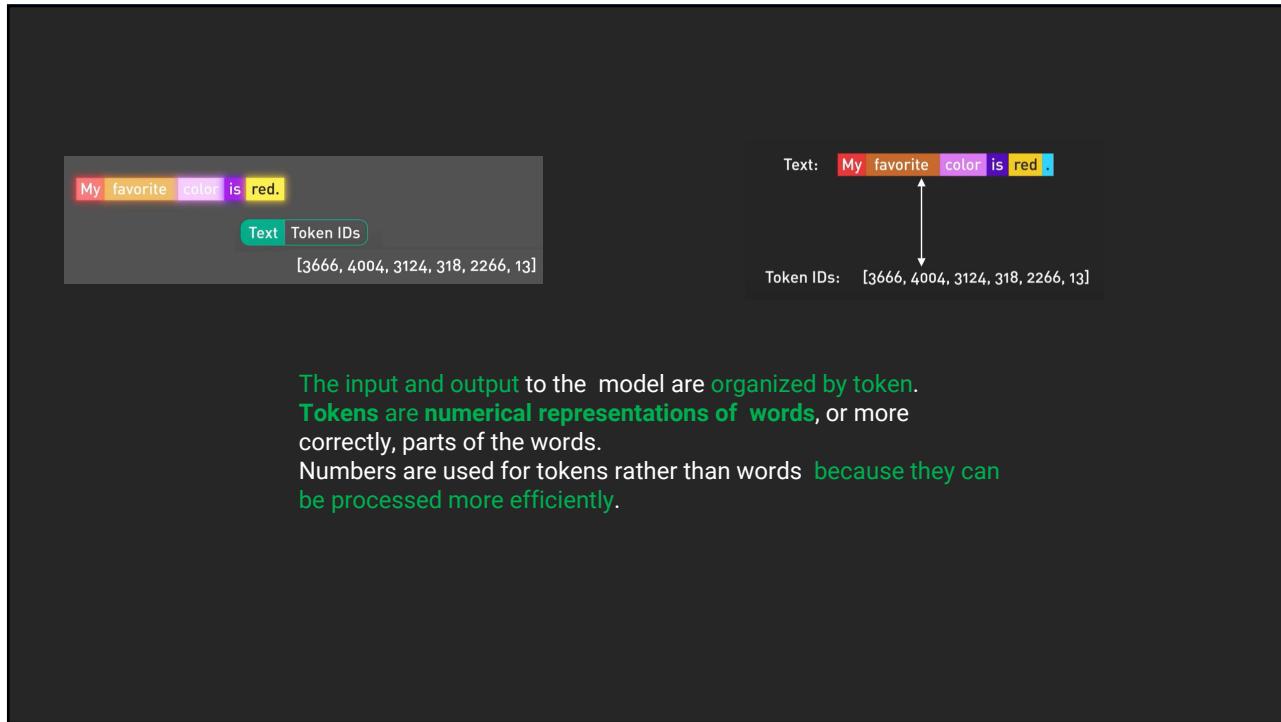


GPT stands for: Generative Pre-trained Transformer

Transformer is a **novel neural network** which is an **attention based encoder decoder type architecture**.

- **First Step: encoding and decoding**
 - When you enter in an input like “**How do I start learning to code**” the input will go through two steps encoding and decoding and these two steps are what make up the Transformer model. **The aim of this step is to try reduce the input into less words** for example “**How learn code**” it’s not grammatically correct but it conveys the same meaning as the a prompt each word is then given some mathematical values so that a computer can process them and by simplifying this, this helps the model **work out** what is the **context** and the **true meaning of the prompt**.
- **Second Step: Attention technique**
 - It is used to **align the words** and to **learn about gender, word order, plurality, etc.**
- **Third Step: Self-attention technique**
 - Then it uses a technique called **self-attention**. This is basically a complex mathematical formula to **work out the importance of each word to the meaning of the sentence** in this case the word **code** is the **most important token**. Moreover, This technique is to define the **exact meaning** of words considering the **context of the text**.

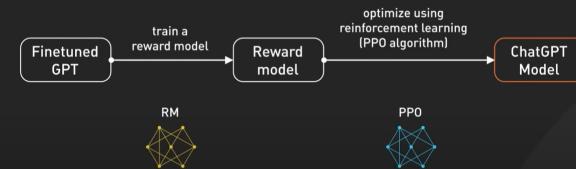




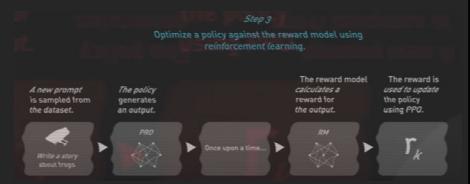
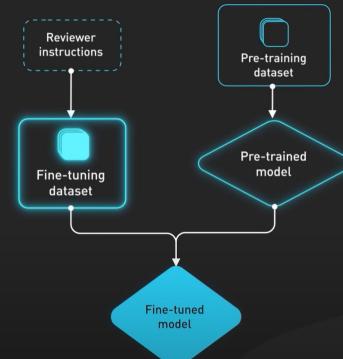
To make the model **safier**, and be **capable of question and answer** in the style of a **chatbot**, the model is further **fine-tuned** to become a version that was used in ChatGPT.

Fine-tuning is a process that turns the model that does not quite align with human values into a fine-tuned model that ChatGPT could use.

This process is called **Reinforcement Training from Human Feedback (RLHF)**.



GPT-3.5 is fine-tuned with RLHF by gathering feedback from people, creating a **reward model** based on their **preferences**, and then iteratively improving the model's performance using **PPO**. This allows GPT-3.5 to generate better responses tailored to specific user requests.



How Open AI ran RLHF on the model:

Imagine GPT-3.5 as a highly skilled chef who can prepare a wide variety of dishes. Fine-tuning GPT-3.5 with RLHF is like refining this chef's skills to make their dishes more delicious.

Initially, the chef is trained with a large dataset of recipes and cooking techniques. However, sometimes the chef doesn't know which dish to make for a specific customer request.

To help with this, we collect feedback from real people to create a new dataset.

The first step is to create a comparison dataset. We ask the chef to prepare multiple dishes for a given request, and then have people rank the dishes based on taste and presentation.

This helps the chef understand which dishes are preferred by the customers.

The next step is reward modeling.

The chef uses this feedback to create a "reward model," which is like a guide for understanding **customer preferences**.

The higher the reward, the better the dish.

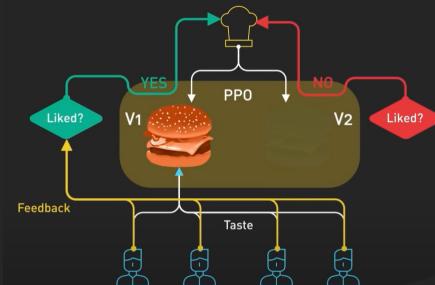
Next, we train the model with PPO, or Proximal Policy Optimization.

In this analogy, the chef practices making dishes while following the reward model. They use a technique called Proximal Policy Optimization to improve their skills.

This is like the chef comparing their current dish with a slightly different version, and learning which one is better according to the reward model.

This process is repeated several times, with the chef refining their skills based on updated customer feedback.

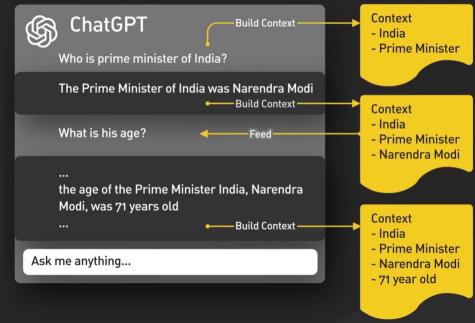
With each iteration, the chef becomes better at preparing dishes that satisfy customer preferences.



How the model is used in ChatGPT to answer a prompt.

How ChatGPT appears to be **context aware**.

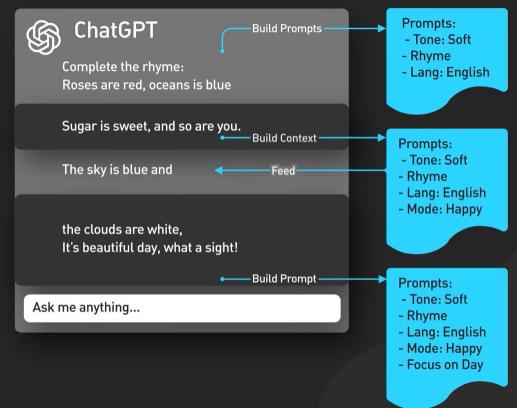
First, ChatGPT **knows the context of the chat conversation**. This is done by ChatGPT UI **feeding the model the entire past conversation** every time a new prompt is entered. This is called **conversational prompt injection**. This is how ChatGPT appears to be context aware.



How the model is used in ChatGPT to answer a prompt.

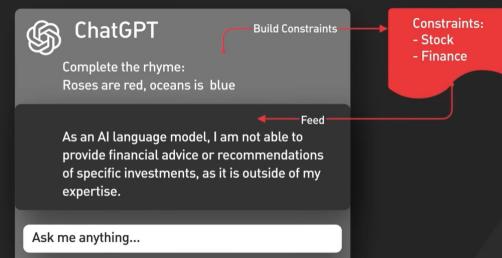
Second, ChatGPT includes **primary prompt engineering**. These are pieces of **instructions injected before and after the user's prompt to guide the model for a conversational tone**.

These prompts are invisible to the user.



How the model is used in ChatGPT to answer a prompt.

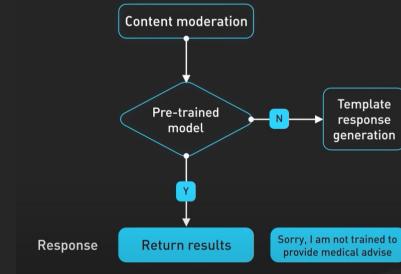
Third, the prompt is passed to the **moderation API** to warn or **block certain types of unsafe content**.



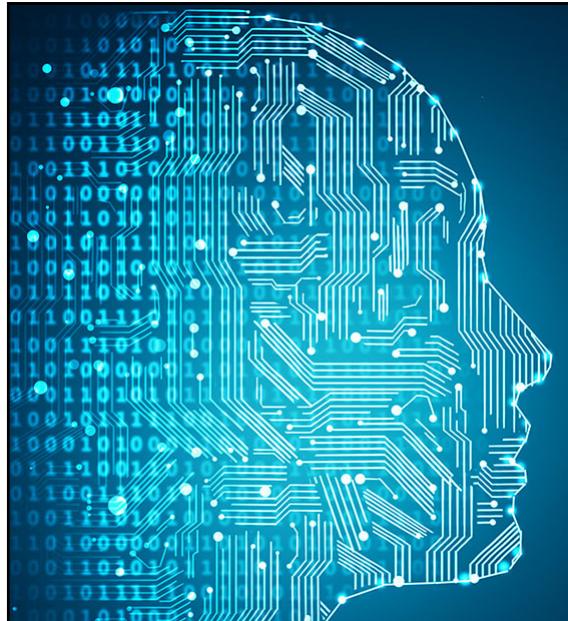
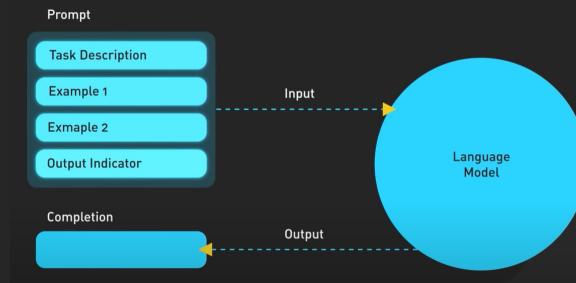
Note: An API (Application Programming Interface) is like a bridge that helps different software applications talk to each other. For example, Apps like Uber use Google Maps' API to show directions and calculate distances.

How the model is used in ChatGPT to answer a prompt.

The generated result is also likely to be passed to the moderation API before returning to the user.



- Prompt engineering



Generative AI for every one
Part 3

Generative AI DeepSeek

The outstanding features

The DeepSeek startup spent about **2.8 million GPU hours** on training their language model. That would take one GPU more than 300 years to do this alone. But of course, they used about 2,000 GPUs so it took them about 2 months to train their model.

The largest Llama 3 model was trained for almost **31 million GPU hours**.

That's about 11 times more than DeepSeek!

Because of the US export regulations, Nvidia cannot currently sell their most powerful GPUs to China. So, to bypass this problem, Nvidia nerfed (weakened) their chips and instead of H100s China now gets less powerful H800s. And these nerfed versions seem cheaper to run, apparently only \$2 per hour, which gets us to this **cost of 5.6 million dollars**. That's orders of magnitude less than the training cost for models like GPT and Gemini.

And the thing is that until now, only these big players like OpenAI, Google, and Meta could be **training such large models due to the costs**, and the rest of us had to wait, and that might change because of DeepSeek now.

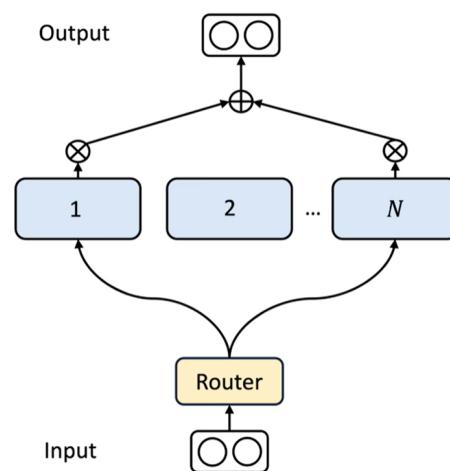
So we have **much less training time**, on **cheaper GPUs**, and the DeepSeek models are also dominating the benchmarks.

There are actually a few **caveats** to these numbers, like the actual cost of training, which is higher than this.

One of the **major changes** that made V3 so much faster is that it uses something called "**Mixture-of-Experts**", or MoE.

The idea behind MoE is that instead of processing input by a single large model, we divide the model into smaller networks, called experts, where each expert specializes in something unique and when we process the input, we don't pass it to all experts. Instead, we use an additional smaller network, called router, which looks at the given input and picks a set of experts and so on.

MoE is applied on every layer of the **transformer network**, and it actually replaces all dense layers. First of all, we're processing the input in these specialized sub-networks that can do the task better. And second, and that's even more important here, we're saving a lot of computation. Because all other experts are being completely ignored.

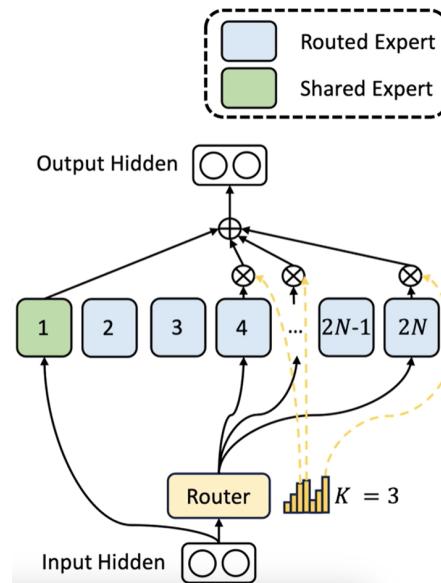


MoE as a concept is nothing new. There are some problems here.

1. How can you actually separate the knowledge and make sure that each expert is specialized?
2. There is a well-known problem called "routing collapse", where the router learns to use only one or very few experts every single time and then ignores the rest. So, you need to somehow encourage the router to select

Deepseek implemented several improvements here.

First, they increase the number of experts and they also added additional shared experts, which are always selected. And the idea behind this is that these shared experts will learn the general knowledge across different contexts, which will leave the other non-shared experts to learn more specialized knowledge. These ideas are nothing novel and have been implemented in NLP many times before, but it's quite impressive that they've got it working on such a large scale.



The DeepSeek website mentions their model having a total of 671 billion parameters but only 37 billion of them are activated. That simply means that for each input token, the network selects only a specific number of experts and the total number of parameters across the selected experts is always 37 billion. However, that doesn't mean you can just load 37 billion parameters on your GPU. You still need to load the entire model if you want to use it because as you generate tokens, you'll be using different sets of experts each time and will eventually need all of them.

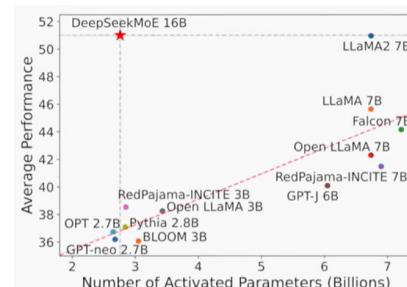
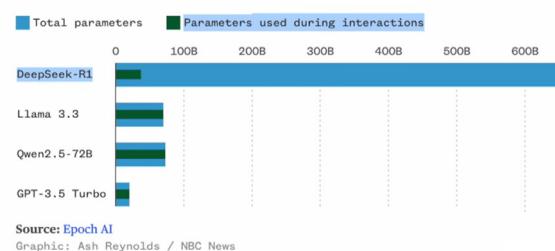


Figure 1 | Comparison between DeepSeekMoE 16B and open source models on the Open LLM Leaderboard. The red dashed line is linearly fitted from data points of all models except DeepSeekMoE 16B. DeepSeekMoE 16B consistently outperforms models with a similar number of activated parameters by a large margin, and achieves comparable performance with LLaMA2 7B, which has approximately 2.5 times the activated parameters.

Source: DeepSeek MoE: Towards Ultimate Expert Specialization in Mixture-of-Experts Language Models, D. Dai, 2024.

The second model released by DeepSeek is called R1. and, R1 is a reasoning model, similar to OpenAI's o1 or o3. The difference between non-reasoning models and reasoning models is that when we use a non-reason model like Llama, it is like spitting out the answer right away. Whereas the reasoning model will first generate lots of reasoning text, which gives the model space and time to self-reflect and explore different reasoning paths. And the final answer is generated at the end. But, OpenAI hides this internal reasoning output from us.

So, it's harder to understand how it works, verify the reasoning, and also to replicate OpenAI's work. And DeepSeek is the first company that figured it out! They published a model that matches the performance of the OpenAI's reasoning models.

One way of learning this complex reasoning behavior would be to train the model on examples of such reasoning and use it for supervised learning. But, that would be quite expensive! So, instead, we only reinforcement learning.

By using reinforcement learning we lead the model to generate lots of solutions for a given problem, and then they used a rule-based reward system to identify correct answers and correct reasoning steps, and then they reinforced those outputs so that the model is more likely to generate them again.

The DeepSeek results are very close to OpenAI's reasoning models. But, their conclusion is that this model still struggles with challenges like poor readability and language mixing. So, they actually decided to collect a small amount of high-quality data and use it to fine-tune the V3 model first before reinforcement learning.

Model	AIME 2024		MATH-500		GPQA		LiveCode Bench		CodeForces	
	pass@1	cons@64	pass@1	pass@1	Diamond	pass@1	pass@1	Bench	pass@1	rating
OpenAI-o1-mini	63.6	80.0	90.0	60.0	53.8	1820				
OpenAI-o1-0912	74.4	83.3	94.8	77.3	63.4	1843				
DeepSeek-R1-Zero	71.0	86.7	95.9	73.3	50.0	1444				

Table 2 | Comparison of DeepSeek-R1-Zero and OpenAI o1 models on reasoning-related benchmarks.

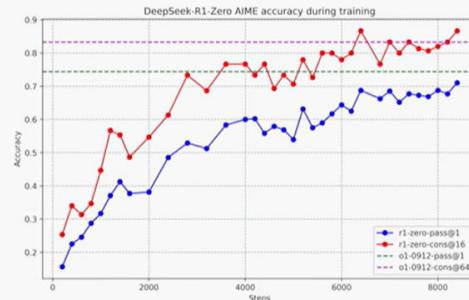


Figure 2 | AIME accuracy of DeepSeek-R1-Zero during training. For each question, we sample 16 responses and calculate the overall average accuracy to ensure a stable evaluation.



Generative AI for every one
Part 3

Generative AI and Business

Task analysis of jobs

Identifying automation opportunities

- AI doesn't automate jobs. It automates tasks.
- Most jobs involve a collection of many tasks.
- Example: Customer service representative

Tasks	Generative AI potential
Answer inbound phone calls from customers	Low
Answer customer chat queries	High
Check status of customer orders	Medium
Keep records of customer interactions	High
Assess accuracy of customer complaints	Low
[...]	[...]

Augmentation vs. Automation

Augmentation

- Help humans with a task.
- E.g., recommend a response for a customer service agent to edit/approve.

Automation

- Automatically perform a task
- E.g., automatically transcribe and summarize records of customer interactions

For some tasks, businesses will start with augmentation, and gradually move toward automation.

Evaluating AI Potential

The potential for augmenting/automating a task depends on:
 (i) Technical feasibility and (ii) Business value.

Technical feasibility: Can AI do it?

- Can a fresh college graduate following the instructions in a prompt complete the task?
- If unsure, try prompting an LLM to see if you can get it to do it.
- An AI engineer can also help assess if RAG, fine-tuning, or other techniques can help.

Business value: How valuable is it for AI to augment or automate this task?

- How much time is spent on this task?
- Does doing this task significantly faster, cheaper or more consistently create substantial value?



Generative AI for every one
Part 3

Generative AI and Business

Additional Job Analysis
examples

In order to distinguish what are the tasks in your work and where generative AI may be able to help, or for the business you may be involved in, to think about how generative AI could help many different tasks in that business, and more specifically performing augmentation or automation on the specified task, we should first analyze the tasks.

Computer Programmer

Tasks	Generative AI potential
Write code	Medium
Write documentation	High
Respond to user support requests	Medium
Review others' code	Low
Gather requirements	Low
[...]	[...]

Lawyer

Tasks	Generative AI potential
Draft and review legal documents	High
Interpret laws, rulings, and regulations	High
Review evidence	Low
Negotiate settlements of disputes	Low
Represent clients in court	Low
[...]	[...]

Landscaper

Tasks	Generative AI potential
Maintain and care for plants	Low
Purchase and transport plants	Low
Maintain yard equipment	Low
Communicate with clients	Medium
Maintain business website	Low
[...]	[...]