

# ChatGPT (GPT- 4) – A Generative Large Language Model

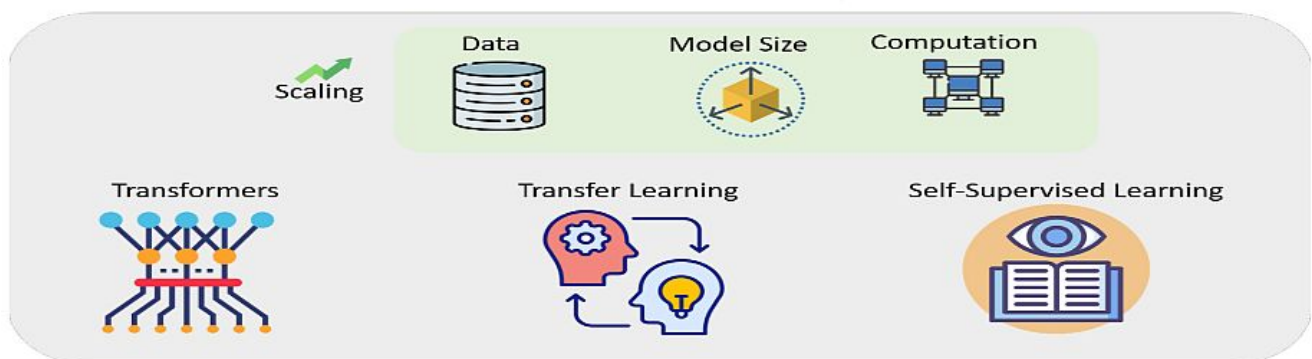
AI tools have evolved and today they can generate completely new texts, codes, images, and videos. ChatGPT, within a short period, has emerged as a leading exemplar of generative artificial intelligence systems.

The results are quite convincing, as it is often hard to recognize whether the content is created by a man or a machine. Generative AI is especially good and applicable in 3 major areas: text, images, and video generation.

## Large Language Models

Text generation as a tool is already being applied in journalism (news production), education (production and misuse of materials), law (drafting contracts), medicine (diagnostics), science (search and generation of scientific papers), etc.

### Large Language Models (LLMs)



Large Language Models – [Source](#)

In 2018, [OpenAI](#) researchers and engineers published an [original work](#) on AI-based generative large language models. They pre-trained the models with a large and diverse corpus of text, in a process they call Generative Pre-Training (GPT).

The authors described how to improve language understanding performance in NLP by using GPT. They applied generative pre-training of a language model on a diverse corpus of unlabeled text, followed by discriminative fine-tuning on each specific task. This annulates the need for human supervision and for [time-intensive hand-labeling](#).

GPT models are based on a [transformer-based](#) deep learning neural network architecture. Their applications include various Natural Language Processing (NLP) tasks, including question answering, text summarization, [sentiment analysis](#), etc. without supervised pre-training.

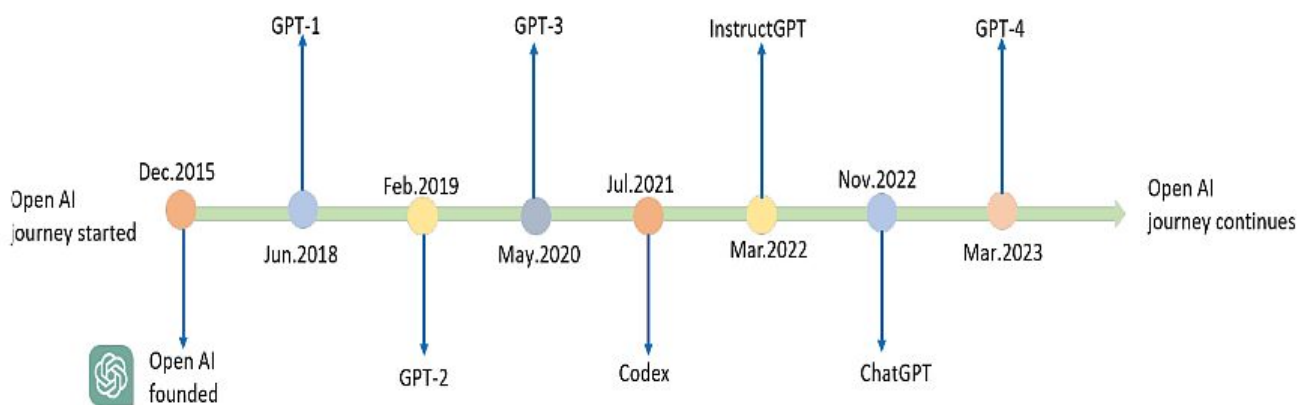
## Previous ChatGPT Models

The GPT-1 version was introduced in June 2018 as a method for language understanding by using generative pre-training. To demonstrate the success of this model, OpenAI refined it and released GPT-2 in February 2019.

The researchers trained GPT-2 to predict the next word based on 40GB of text. Unlike other AI models and practices, OpenAI did not publish the full version of the model, but a lite version. In July 2020, they introduced the GPT-3 model as the most advanced language model with 175 billion parameters.

## *GPT-2 Model*

GPT-2 model is an [unsupervised multi-task learner](#). The advantages of GPT-2 over GPT-1 were using a larger dataset and adding more parameters to the model to learn stronger language models. The training objective of the language model was formulated as  $P(\text{output}|\text{input})$ .



GPT Models Timeline – [Source](#)

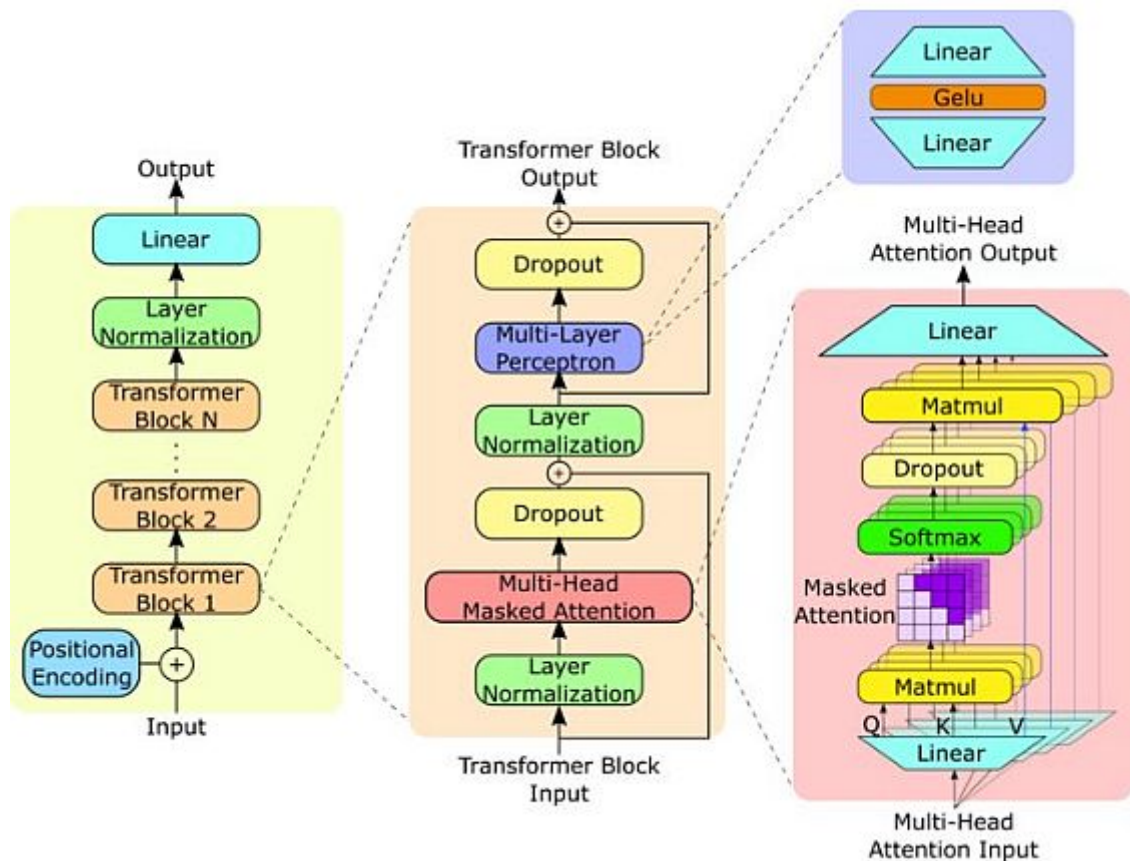
GPT-2 is a large transformer-based language model, trained to predict the next word in a sentence. The transformer provides a mechanism based on encoder-decoders to detect input-output dependencies. Today, it is the golden approach for generating text.

You don't need to train GPT-2 (it is already pre-trained). GPT-2 is not just a language model like BERT, it can also generate text. Just give it the beginning of the phrase upon typing, and then it will complete the text word by word.

At first, recurrent ([RNN](#)) networks, in particular, LSTM, were mainstream in this area. But after the invention of the Transformer architecture in the summer of 2017 by OpenAI, GPT-2 gradually began to prevail in conversational tasks.

## *GPT-2 Model Features*

To improve the performance, in February 2019, OpenAI increased its GPT by 10 times. They trained it on an even larger volume of text, on 8 million Internet pages (a total of 40 GB of text).



GPT-2 Model Architecture – [Source](#)

The resulting GPT-2 network was the [largest neural network](#), with an unprecedented number of 1.5 billion parameters. Other features of GPT-2 include:

- GPT-2 had 48 layers and used 1600-dimensional vectors for word embedding.
- Large vocabulary of 50,257 tokens.
- Larger batch size of 512 and a larger context window of 1024 tokens.
- Researchers performed [normalization](#) at the input of each sub-block. Moreover, they added a layer after the final self-attention block.





As a result, GPT-2 was able to generate entire pages of connected text. Also, it reproduced the names of the characters in the course of the story, quotes, references to related events, and so on.

Generating coherent text of this quality is impressive by itself, but there is something more interesting here. GPT-2 without any additional training immediately showed results close to the state-of-the-art on many conversational tasks.

## GPT-3

GPT-3 release took place in May 2020 and beta testing began in July 2020. All three GPT generations utilize [artificial neural networks](#). Moreover, they train these networks on raw text and [multimodal](#) data.

At the [heart of the Transformer](#) is the attention function, which calculates the probability of occurrence of a word depending on the context. The algorithm learns contextual relationships between words in the texts provided as training examples and then generates a new text.

 GPT-3	ada <sup>Raw</sup> text-ada <sup>SFT</sup>	babbage <sup>Raw</sup> text-babbage <sup>SFT</sup>	curie <sup>Raw</sup> text-curie <sup>SFT</sup>	davinci <sup>Raw</sup> text-davinci <sup>SFT</sup>
 GPT-3.5	code-davinci-002 <sup>Code</sup>	text-davinci-002 <sup>SFT</sup>	text-davinci-003 <sup>RLHF</sup>	
 ChatGPT	gpt-3.5-turbo <sup>RLHF-Chat</sup>	gpt-3.5-turbo-16k <sup>RLHF-Chat</sup>		
 GPT-4	gpt-4 <sup>RLHF-Chat</sup>	gpt-4-32k <sup>RLHF-Chat</sup>		

GPT-3 Models Evolution (SFT-supervised fine-tuning, RLHF-reinforcement learning from human feedback) – [Source](#)

- GPT-3 shares the same architecture as the previous GPT-2 algorithm. The main difference is that they increased the number of parameters to 175 billion. Open-AI trained GPT-3 on 570 gigabytes of text or 1.5 trillion words.
- The training materials included: the entire Wikipedia, two datasets with books, and the second version of the WebText dataset.
- The GPT-3 algorithm can create texts of different forms, styles, and purposes: journal and book stories (while imitating the style of a particular author), songs and poems, press releases, and technical manuals.
- OpenAI tested GPT-3 in practice where it wrote multiple journal essays (for the UK news magazine Guardian). The program can also solve anagrams, solve simple arithmetic examples, and generate tablatures and computer code.

## ChatGPT – The Latest GPT-4 Model

OpenAI launched its latest version, the GPT-4 model, on March 14, 2023, together with its publicly available ChatGPT bot, and sparked an AI revolution.

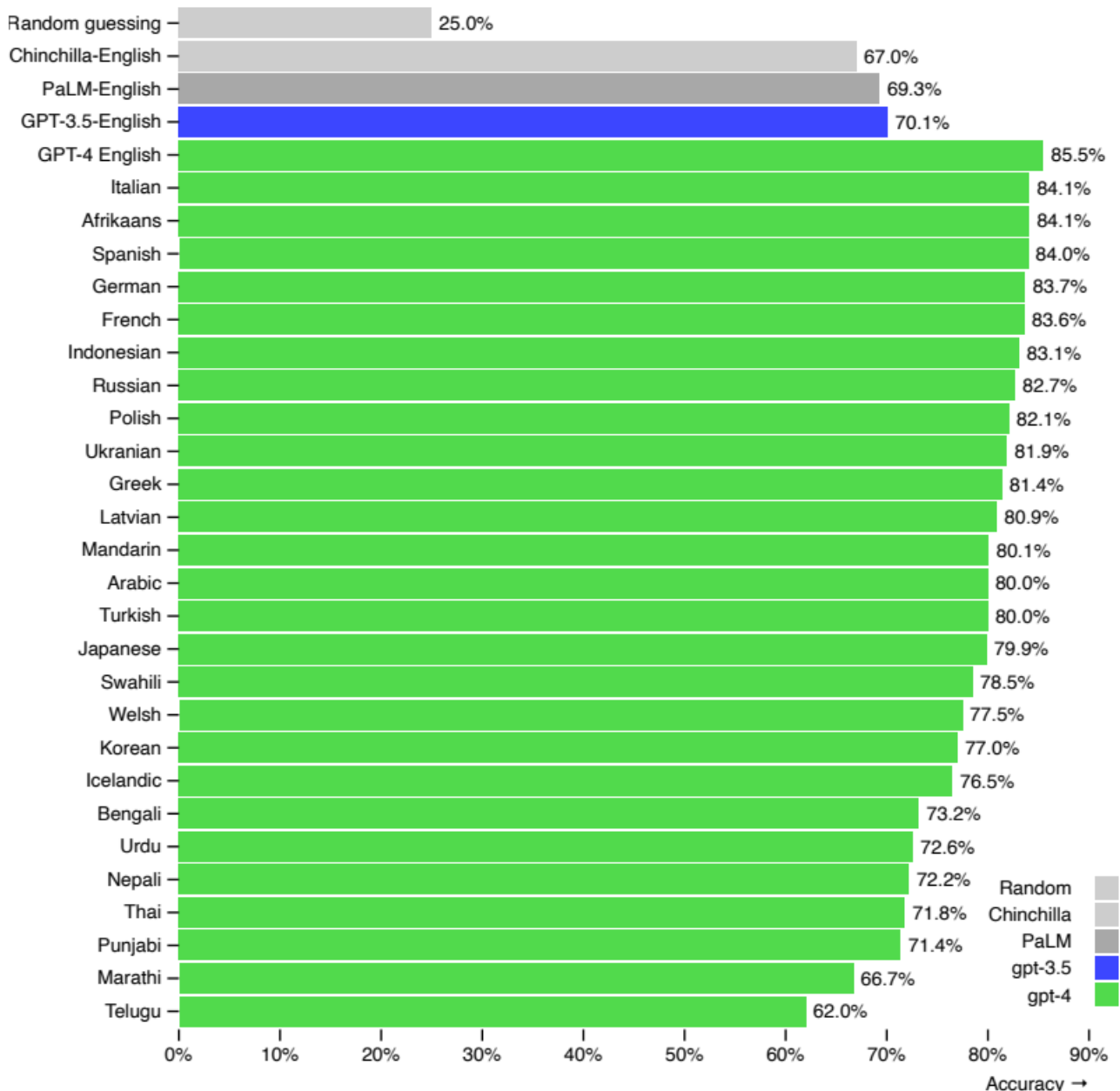
### GPT-4 New Features

If we compare Chat GPT 3 vs 4, the new model [processes images](#) and text as input, something that previous versions could only do with text.

The new version has increased API tokens from 4096 to 32,000 tokens. This is a major improvement, as it provides the creation of increasingly complex and specialized texts and conversations. Also, GPT-4 has a larger training set volume than GPT-3, i.e. up to 45 TB.

OpenAI trained the model on a large amount of multimodal data, including images and text from multiple domains and sources. They sourced data from various public datasets, and the objective is to predict the next token in a document, given a sequence of previous tokens and images.

## GPT-4 3-shot accuracy on MMLU across languages



ChatGPT MMLU (Massive Multitask Language Understanding) accuracy – [Source](#)

- In addition, GPT-4 improves problem-solving capabilities by offering greater responsiveness with text generation that imitates the style and tone of the context.
- New knowledge limit: the message that the information collected by ChatGPT has a cut-off date of September 2021 is coming to an end. The new model includes information up to April 2023, providing a much more current query context.
- Better instruction tracking: The model works better than previous models for tasks that require careful tracking of instructions, such as generating specific formats.
- Multiple tools in a chat: the updated GPT-4 chatbot chooses the appropriate tools from the drop-down menu.

## ChatGPT Performance

GPT-4 (ChatGPT) exhibits [human-level performance](#) on the majority of professional and academic exams. Notably, it passes a simulated version of the Uniform Bar Examination with a score in the top 10% of test takers.

	GPT-4 Evaluated few-shot	GPT-3.5 Evaluated few-shot	LM SOTA Best external LM evaluated few-shot	SOTA Best external model (incl. benchmark-specific tuning)
<b>MMLU [49]</b> Multiple-choice questions in 57 subjects (professional & academic)	<b>86.4%</b> 5-shot	<b>70.0%</b> 5-shot	<b>70.7%</b> 5-shot U-PaLM [50]	<b>75.2%</b> 5-shot Flan-PaLM [51]
<b>HellaSwag [52]</b> Commonsense reasoning around everyday events	<b>95.3%</b> 10-shot	<b>85.5%</b> 10-shot	<b>84.2%</b> LLaMA (validation set) [28]	<b>85.6</b> ALUM [53]
<b>AI2 Reasoning Challenge (ARC) [54]</b> Grade-school multiple choice science questions. Challenge-set.	<b>96.3%</b> 25-shot	<b>85.2%</b> 25-shot	<b>85.2%</b> 8-shot PaLM [55]	<b>86.5%</b> ST-MOE [18]
<b>WinoGrande [56]</b> Commonsense reasoning around pronoun resolution	<b>87.5%</b> 5-shot	<b>81.6%</b> 5-shot	<b>85.1%</b> 5-shot PaLM [3]	<b>85.1%</b> 5-shot PaLM [3]
<b>HumanEval [43]</b> Python coding tasks	<b>67.0%</b> 0-shot	<b>48.1%</b> 0-shot	<b>26.2%</b> 0-shot PaLM [3]	<b>65.8%</b> CodeT + GPT-3.5 [57]
<b>DROP [58] (F1 score)</b> Reading comprehension & arithmetic.	<b>80.9</b> 3-shot	<b>64.1</b> 3-shot	<b>70.8</b> 1-shot PaLM [3]	<b>88.4</b> QDGAT [59]
<b>GSM-8K [60]</b> Grade-school mathematics questions	<b>92.0%*</b> 5-shot chain-of-thought	<b>57.1%</b> 5-shot	<b>58.8%</b> 8-shot Minerva [61]	<b>87.3%</b> Chinchilla + SFT+ORM-RL, ORM reranking [62]

GPT-4 Performance on Academic Benchmarks – [Source](#)

The model's capabilities on bar exams originate primarily from the pre-training process, and they don't depend on RLHF. On multiple-choice questions, both the base GPT-4 model and the RLHF model perform equally well.

On a dataset of 5,214 prompts submitted to ChatGPT and the OpenAI API, the responses generated by GPT-4 were better than the GPT-3 responses on 70.2% of prompts.

GPT-4 accepts prompts consisting of both images and text, which lets the user specify any vision or language task. Moreover, the model generates text outputs given inputs consisting of arbitrarily interlaced text and images. Over a range of domains (including images), ChatGPT generates superior content to its predecessors.

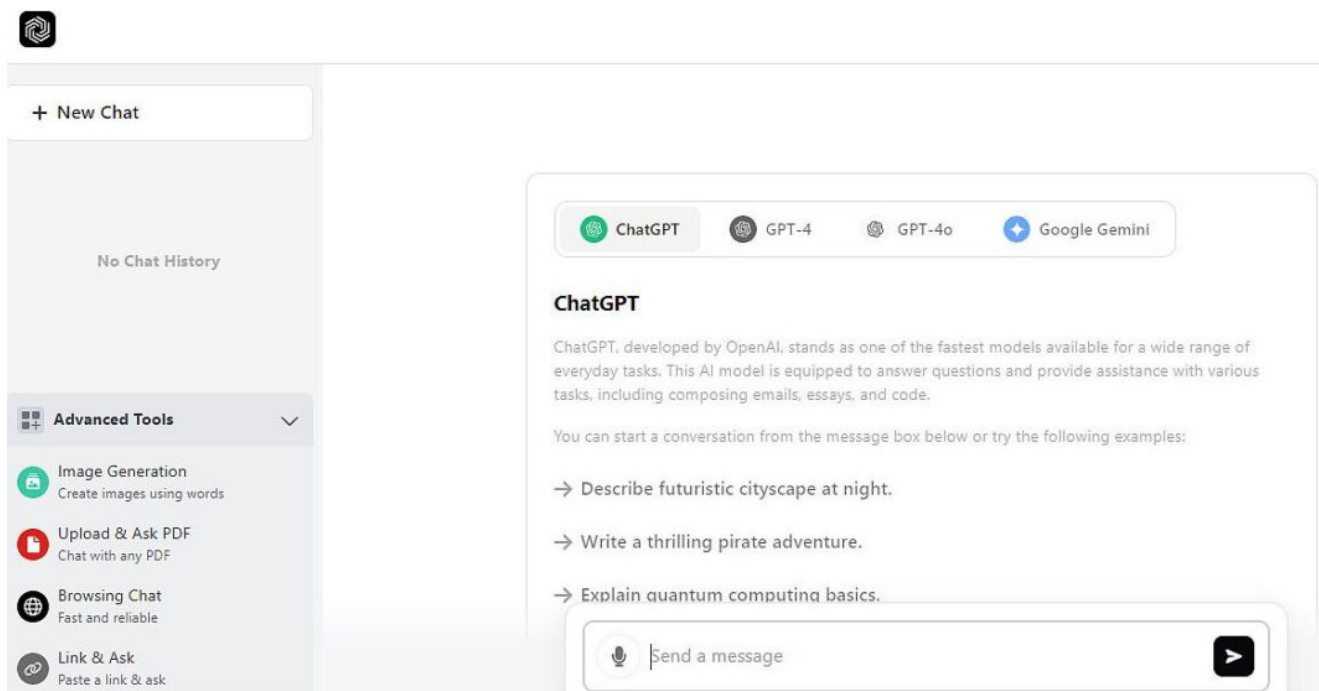
## How to Use ChatGPT 4

You can access ChatGPT [here](#), and its interface is easy and clear. The basic usage is a free version, while the Plus plan costs a \$20 per month subscription. There are also Team and Enterprise plans. For all of them, you need to create an account.

Here are the main ChatGPT 4 options, with the screenshot below as an example:

- **Chat bar and sidebar:** The chat bar “Send a message” button is placed on the bottom of the screen. ChatGPT remembers your previous conversations and will respond with context. When you register and log in, the bot can remember your conversations.





### Using ChatGPT – [Source](#)

- **Account (if registered):** Clicking on your name in the upper right corner gives you access to your account information, including settings, the option to log out, get help, and customize ChatGPT.
- **Chat history:** In Advanced tools (left sidebar), you can access GPT-4 past conversations. You can also share your chat history with others, turn off chat history, delete individual chats, or delete your entire chat history.
- **Your Prompts:** The questions or prompts you send the AI chatbot appear at the bottom of the chat window, with your account details on the top right.
- **ChatGPT's responses:** ChatGPT responds to your queries and the responses appear on the main screen. Also, you can copy the text to your clipboard to paste it elsewhere and provide feedback on whether the response was accurate.

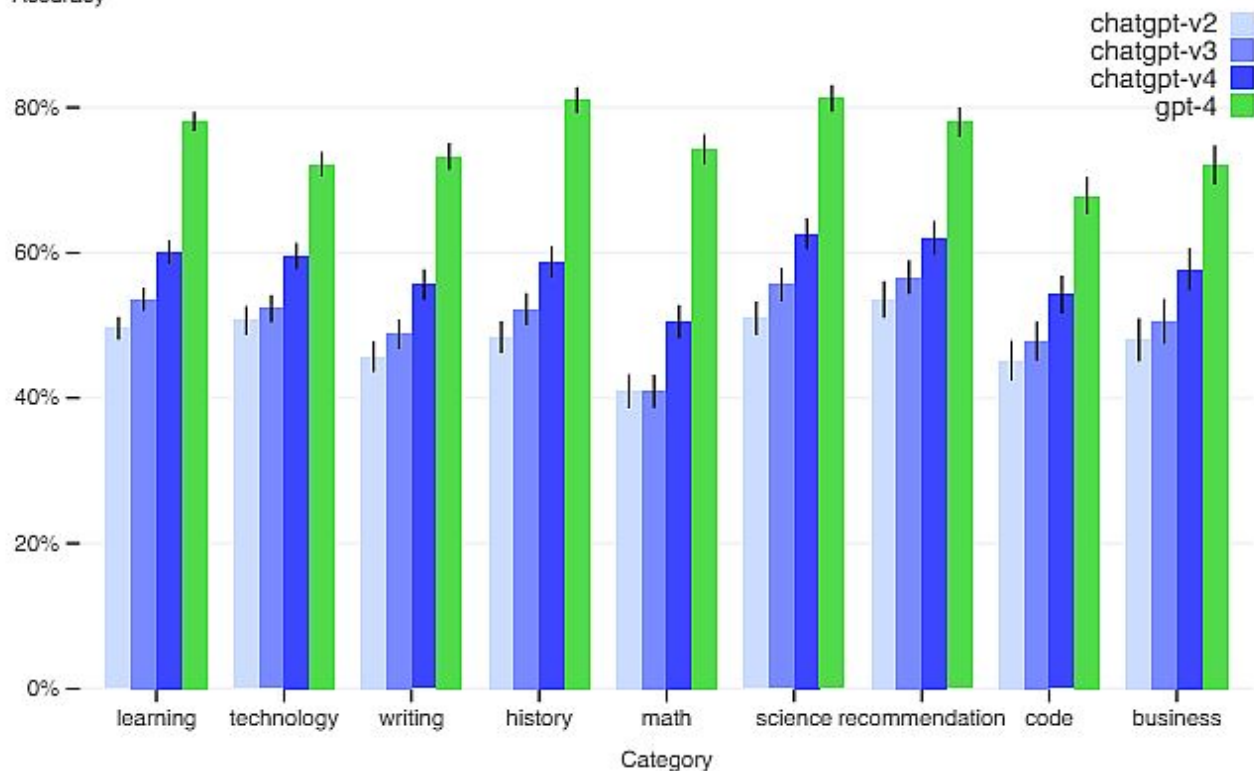
## *Limitations of ChatGPT*

Despite its capabilities, GPT-4 has similar limitations to earlier GPT models. Most importantly, it is still not fully reliable (it “hallucinates” facts). You should be careful when using ChatGPT outputs, particularly in high-stakes contexts, with the exact protocol for specific applications.

GPT-4 significantly reduces hallucinations relative to previous GPT-3.5 models (which have themselves been improving with continued iteration). Thus, GPT-4 scores 19 percentage points higher than the previous GPT-3.5 on OpenAI evaluations.

## Internal factual eval by category

Accuracy



ChatGPT Performance Limitations by Category – [Source](#)

GPT-4 generally lacks knowledge of events that have occurred after the pre-training data cuts off on September 10, 2021, and does not learn from its experience. It can sometimes make simple reasoning errors that do not seem to comport with competence across so many domains.

Also, GPT-4 can be confidently wrong in its predictions, not double-checking the output when it's likely to make a mistake. GPT-4 has various biases in its outputs that OpenAI still tries to characterize and manage.

Open AI intends to make GPT-4 have reasonable default behaviors that reflect a wide swath of users' values. Therefore, they will customize their system within some broad bounds and get public feedback on improving it.

## What's Next?

ChatGPT is a large multimodal model capable of processing image and text inputs and producing text outputs. The model can be used in a wide range of applications, such as dialogue systems, text summarization, and machine translation. As such, it will be the subject of substantial interest and progress in the upcoming years.

Is this blog interesting? Read more of our similar blogs here:

- [Modality: The Multi-Dimensional Language of Computer Vision](#)
  - [Large Action Models: Beyond Language, Into Action](#)
  - [Evolution of Motion Tracking](#)
  - [Llama 2: The Next Revolution in AI Language Models](#)
  - [CLIP: Contrastive Language-Image Pre-Training](#)
-



# GPT-2 Detailed Model Architecture

Here a detailed architectural diagram of GPT-2 that shows how input data transforms as it flows through the model. The diagram is meant to help you trace the steps the model takes from input to output, while a brief explanation of how text is prepared before it's fed into the model.

## Text Preparation

Before we get to the diagram, let's first understand how text input is prepared. The raw text has to be tokenized, which means converting words into integers that map to indices in the model's vocabulary. GPT-2 uses a process called [Byte-Pair Encoding](#) (BPE) for tokenization, which breaks text down into subwords.

We will use [Tiktoken](#), a fast BPE tokenizer used in OpenAI's models. Here is an example:

```
1 import tiktokenenc = tiktoken.get_encoding("gpt2") text = """In a remote mountain village,
   nestled among the towering peaks and pine forests, \a solitary storyteller sat by a crackling
   bonfire. Their voice rose and fell like a melodic river, \weaving tales of ancient legends and
   forgotten heroes that captivated the hearts of the villagers gathered around. \The stars above
   shone brightly, their twinkling adding a celestial backdrop to the mesmerizing stories, \as
   the storyteller transported their audience to worlds of wonder and imagination. """tokens =
   enc.encode(text) tokens.append(enc.eot_token) print(tokens[:10])
```

```
1 [818, 257, 6569, 8598, 7404, 11, 16343, 992, 1871, 262]
```

This turns the text into a list of integers representing subwords or words from GPT-2's vocabulary. We also append a special `<|endoftext|>` token to mark the end of the text sequence.

Next, we will rearrange the tokens into a format the GPT model can process. The model expects the input data to be in the form of a batch of sequences. The input tensor should have the shape **(B, T)**, where:

- B is the batch size (how many sequences we process in parallel).
- T is the sequence length (number of tokens in each sequence).

To demonstrate, we will create a batch of 5 sequences, each containing 10 tokens, using the first 50 tokens of the input:

```
1 import torchB, T = 5, 10data = torch.tensor(tokens[:50+1])x = data[:-1].view(B, T) y =
   data[1:].view(B, T) print(x)print(y)
```

```
1 tensor([[ 818,  257, 6569, 8598, 7404,  11, 16343,  992, 1871,  262],
          [25740,  290, 20161, 17039,  11,  257, 25565, 1621,  660],
          [ 6051, 3332,  416, 257, 8469, 1359, 5351, 6495,  13, 5334],
          [ 3809, 8278,  290, 3214,  588, 257, 7758, 29512, 7850,  11],
          [44889, 19490,  286, 6156, 24901,  290, 11564, 10281,  326, 3144]])
   tensor([[ 257, 6569, 8598, 7404,  11, 16343,  992, 1871,  262, 38879],
          [25740,  290, 20161, 17039,  11,  257, 25565, 1621,  660, 6051],
          [3332,  416,  257, 8469, 1359, 5351, 6495,  13, 5334, 3809],
          [ 8278,  290, 3214,  588, 257, 7758, 29512, 7850,  11, 44889],
          [19490,  286, 6156, 24901, 290, 11564, 10281,  326, 3144, 30829]])
```

Here, `x` contains the input tokens, and `y` contains the target tokens, shifted by one position so that the model can learn to predict the next token in the sequence. During training, these tensors are fed into the model, with the target tensor `y` used to calculate the loss using cross-entropy loss.

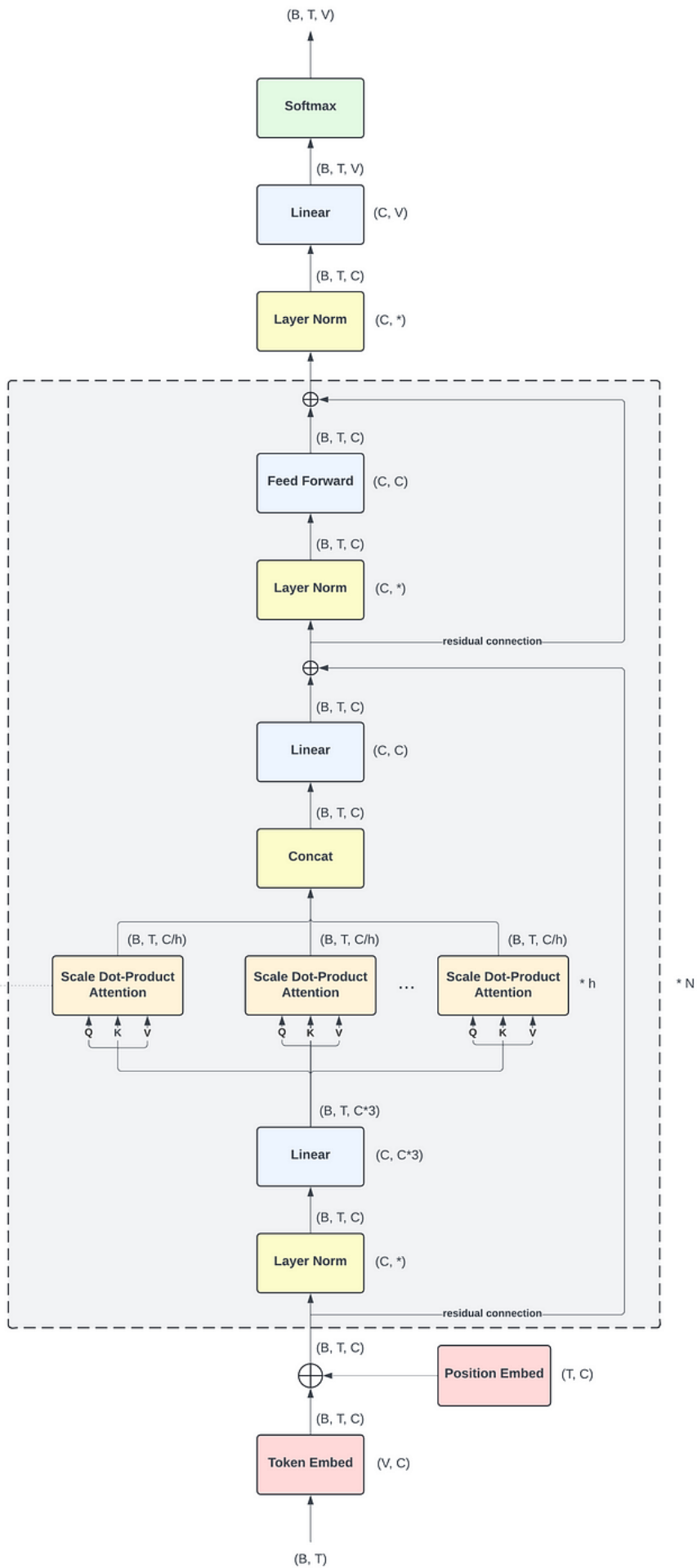
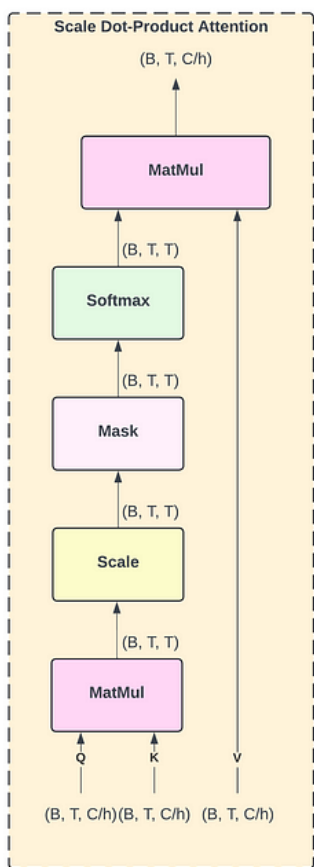
# GPT-2 Architectural Diagram

---

Finally, let's take a look at the GPT-2 architectural diagram to better understand how the input data moves through the model. Below are some key configuration parameters for GPT-2:

- **Vocabulary Size (V):** 50,257
- **Maximum Sequence Length (T):** 1024
- **Embedding Dimensionality (C):** 768
- **Number of Heads (h):** 12
- **Number of Layers (N):** 12
- **Batch Size (B):** 512

The diagram and configuration will help you trace how the data transforms at each stage, from the input to the output.



GPT-2 Architectural Diagram

## GPT-3

---

OpenAI's GPT-3 is the third generation of the autoregressive language model series. Its architecture shares similarities with the traditional transformer architecture, which includes encoders, decoders, and attention layers. Below are the architectural details of the GPT-3 model:

- **Number of Layers:** The largest version of GPT-3, known as GPT-3 175B, has 96 transformer layers.
- **Number of Attention Heads:** Each transformer layer in GPT-3 175B contains 96 attention heads.
- **Hidden Layer Dimensions:** The dimension of the hidden layers in GPT-3 175B is 12,288.
- **Feedforward Dimensions:** In a transformer, the feedforward layer typically sets its dimension to four times the size of the block's output dimension. For GPT-3 175B, the feedforward dimension is 49,152 ( $12,288 \times 4$ ).
- **Token Dimensions:** All GPT-3 models use a token dimension of 2048.
- **Attention Head Dimensions:** Each attention head in GPT-3 has a dimension of 128.
- **Total Number of Parameters:** GPT-3 175B consists of 175 billion trainable parameters.
- **Attention Mechanism:** GPT-3 employs a multi-head self-attention mechanism. Each token in the input attends to every other token, allowing the model to understand context from all parts of the input. The attention mechanism involves query, key, and value matrices. The input embeddings are transformed into these three matrices. The query matrix interacts with the key and value matrices to compute attention scores. The scaled dot-product attention calculates attention scores using the dot product of the query and key matrices. After scaling, these scores are passed through a softmax function to obtain attention weights. Multi-head attention enables the model to focus on different parts of the input simultaneously, enhancing contextual understanding.
- **Positional Encoding:** Unlike recurrent neural networks (RNNs), transformers do not process data sequentially. To capture word order, ChatGPT uses positional encoding to add positional information for each token in the sequence, helping the model better understand context.
- **Sparse Attention Patterns:** GPT-3 utilizes alternating dense and local sparse attention patterns within the transformer, similar to the Sparse Transformer.
- **Batch Size:** The batch size for GPT-3 175B is 3.2 million.

The GPT-3 family comprises eight models of varying sizes, ranging from 125 million parameters to 175 billion parameters. The smaller models have fewer layers and parameters. For example, the smallest GPT-3 model has an architecture similar to BERT, with 12 attention layers, each equipped with 64-dimensional heads ( $12 \times 64$ ). OpenAI trained these models to study the relationship between model size and machine learning performance. Research suggests that with sufficient training data, validation loss can be approximated as a power-law distribution. Training models of different sizes helps test this hypothesis, including the relationship between validation loss and downstream language tasks.

GPT-3's architecture is an optimized and scaled-up version of the GPT-2 architecture, featuring modified initialization, pre-normalization, reverse tokenization, and the use of alternating dense and sparse attention patterns. GPT-3 is trained on a vast corpus of text data, learning to predict the next word in a sentence. This pre-training phase helps the model understand grammar, facts about the world, and some reasoning abilities. After pre-training, the model undergoes fine-tuning on a narrower dataset with human reviewers following specific guidelines to align the model with desired behaviors for specific applications.

The transformer architecture, introduced in the 2017 paper "Attention Is All You Need" by Vaswani et al., revolutionized natural language processing by enabling models to capture long-range dependencies and parallel processing capabilities. The transformer architecture consists of an encoder that processes input sequences and a decoder that generates output. In between, there are multiple attention and feedforward layers. GPT-3 is a decoder-only transformer model, meaning it takes embeddings as input and produces text as output.

The working principle of GPT-3 can be summarized as follows: Input text is tokenized into smaller units called tokens, which are then converted into embeddings. Positional encodings are added to these embeddings to retain sequence information. The tokens are processed through multiple transformer layers. Each layer comprises two main components: the self-attention mechanism, where each token in the input attends to every other token to understand context from all parts of the input, and the feedforward networks, which apply transformations to the attended information to enable the model to learn complex patterns. After passing through the transformer layers, the final hidden states are used to generate output tokens. The model employs a softmax layer to predict the probability distribution over the vocabulary for the next token, generating text step-by-step.

The massive scale of GPT-3 allows it to capture and utilize an unprecedented amount of knowledge during conversations and open-ended tasks. Its training data spans a wide array of domains, including news articles, scientific papers, books, and social media posts. This extensive training data endows the model with broad, almost encyclopedic knowledge, enabling it to fluidly draw upon this knowledge during interactions. Below is a table illustrating the parameters and architectural details of different GPT-3 models:

<b>Model Name</b>	<b>Number of Parameters (in billions)</b>	<b>Number of Layers</b>	<b>Hidden Layer Dimensions</b>	<b>Feedforward Dimensions</b>	<b>Attention Head Dimensions</b>	<b>Number of Attention Heads</b>
GPT-3 175B	175	96	12,288	49,152	128	96
GPT-3 70B	70	56	12,288	49,152	128	96
GPT-3 13B	13	24	12,288	49,152	128	96
GPT-3 3.3B	3.3	16	12,288	49,152	128	96
GPT-3 1.3B	1.3	16	12,288	49,152	128	96
GPT-3 355M	0.355	8	12,288	49,152	128	96
GPT-3 125M	0.125	8	12,288	49,152	128	96.

The GPT-3 model has achieved remarkable performance on a wide range of language tasks. Its few-shot learning capability is particularly notable. With minimal examples, it can adapt to new tasks without fine-tuning. This makes it highly versatile and powerful.

# Understanding the Evolution of ChatGPT: Part 2 – GPT-2 and GPT-3

---

This is the second article of our [Gpt](#) series, where we will dive into the development of GPT-2 and GPT-3, with model size increased from 117M to a staggering 175B.

In case you are interested in the other articles in this GPT series, check the links below:

- Part 1: [Understanding the Evolution of ChatGPT: Part 1 – An In-Depth Look at GPT-1 and What Inspired It.](#)
- Part 3: [Insights from Codex and InstructGPT](#)

We choose to cover GPT-2 and GPT-3 together not just because they share similar architectures, but also they were developed with a common philosophy aimed at bypassing the finetuning stage in order to make LLMs truly intelligent. Moreover, to achieve that goal, they both explored several key technical elements such as task-agnostic learning, scale hypothesis and in-context learning, etc. Together they demonstrated the power of training large models on large datasets, inspired further research into emergent capabilities, established new evaluation protocols, and sparked discussions on enhancing the safety and ethical aspects of LLMs.

Below are the contents we will cover in this article:

- **Overview:** The paradigm shift towards bypassing finetuning, and the three key elements made this possible: task-agnostic learning, the scaling hypothesis, and in-context learning.
  - **GPT-2:** Model architecture, training data, evaluation results, etc.
  - **GPT-3:** Core concepts and new findings.
  - **Conclusions.**
- 

## Overview

---

### *The Paradigm Shift Towards Bypassing Finetuning*

In our [previous article](#), we revisited the core concepts in GPT-1 as well as what had inspired it. By combining auto-regressive language modeling pre-training with the decoder-only Transformer, GPT-1 had revolutionized the field of [NLP](#) and made **pre-training plus finetuning** a standard paradigm.

But OpenAI didn't stop there.

Rather, while they tried to understand why language model pre-training of Transformers is effective, they began to notice the zero-shot behaviors of GPT-1, where as pre-training proceeded, the model was able to steadily improve its performance on tasks that it hadn't been finetuned on, showing that pre-training could indeed improve its zero-shot capability, as shown in the figure below:



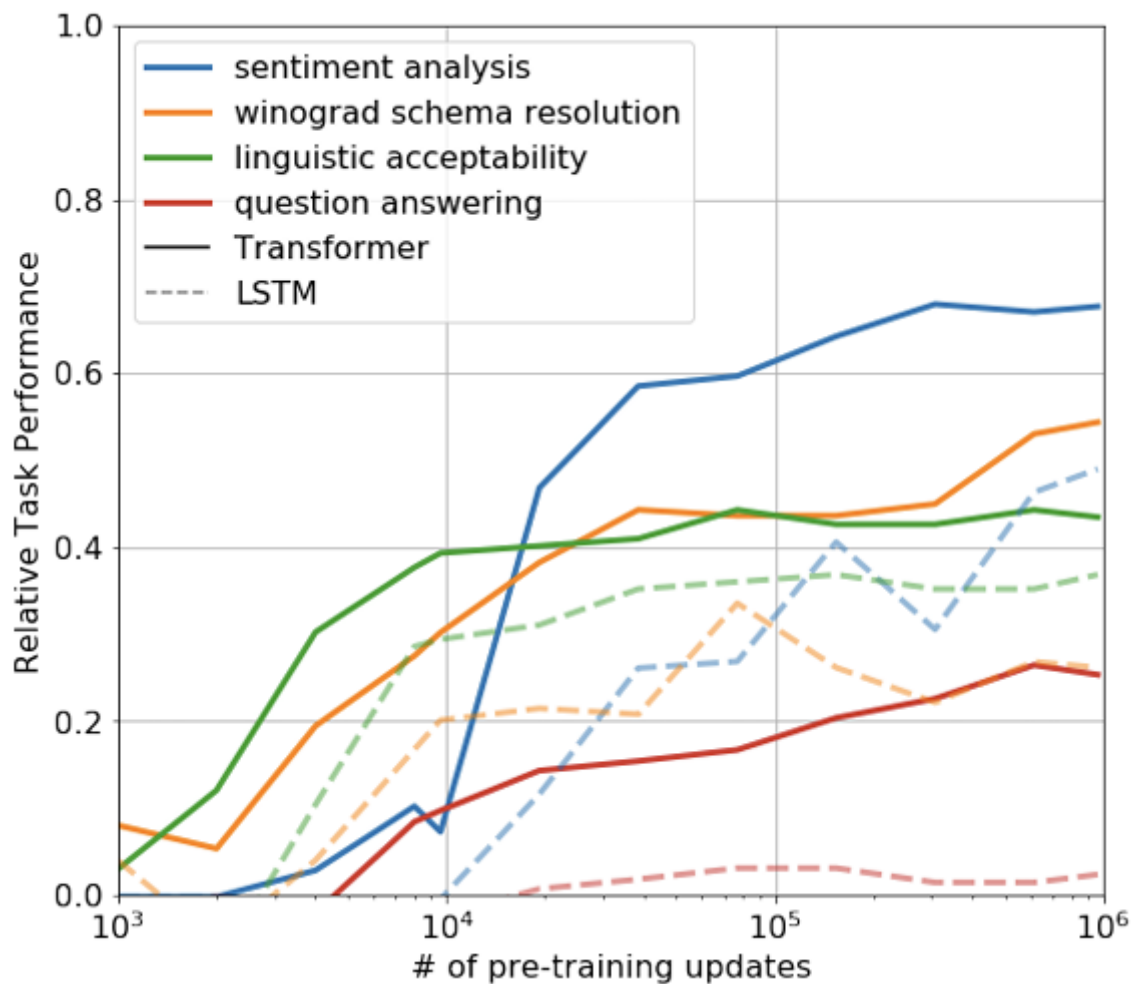


Figure 1. Evolution of zero-shot performance on different tasks as a function of LM pre-training updates. (Image from the [GPT-1 paper](#).)

This motivated the paradigm shift from "**pre-training plus finetuning**" to "**pre-training only**", or in other words, a task-agnostic pre-trained model that can handle different tasks **without finetuning**.

Both GPT-2 and GPT-3 are designed following this philosophy.

But why, you might ask, isn't the **pre-training plus finetuning** magic \*\*\*\* working just fine? What are the additional benefits of bypassing the finetuning stage?

## ***Limitations of Finetuning***

Finetuning is working fine for some well-defined tasks, but not for all of them, and the problem is that there are numerous tasks in the NLP domain that we have never got a chance to experiment on yet.

For those tasks, the requirement of a finetuning stage means we will need to collect a finetuning dataset of meaningful size for each individual new task, which is clearly not ideal if we want our models to be truly intelligent someday.

Meanwhile, in some works, researchers have observed that there is an increasing risk of exploiting spurious correlations in the finetuning data as the models we are using become larger and larger. This creates a paradox: the model needs to be large enough so that it can absorb as much information as possible during training, but finetuning such a large model on a small, narrowly distributed dataset will make it struggle when generalize to out-of-distribution samples.

Another reason is that, as humans we do not require large supervised datasets to learn most language tasks, and if we want our models to be useful someday, we would like them to have such fluidity and generality as well.

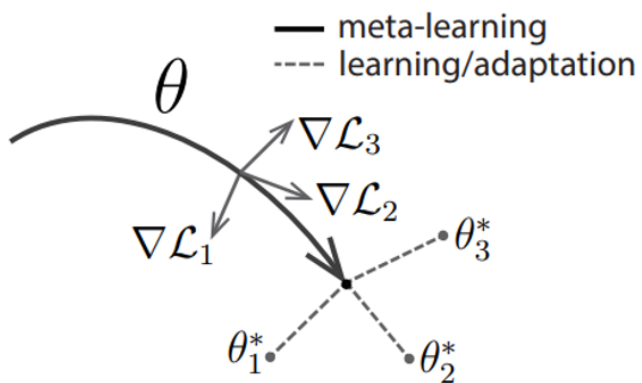
Now perhaps the real question is that, what can we do to achieve that goal and bypass finetuning?

Before diving into the details of GPT-2 and GPT-3, let's first take a look at the three key elements that have influenced their model design: task-agnostic learning, the scale hypothesis, and in-context learning.

## Task-agnostic Learning

Task-agnostic learning, also known as **Meta-Learning** or **Learning to Learn**, refers to a new paradigm in machine learning where the model develops a broad set of skills at training time, and then uses these skills at inference time to rapidly adapt to a new task.

For example, in [MAML](#) (Model-Agnostic Meta-Learning), the authors showed that the models could adapt to new tasks with very few examples. More specifically, during each inner loop (highlighted in blue), the model firstly samples a task from a bunch of tasks and performs a few gradient descent steps, resulting in an adapted model. This adapted model will be evaluated on the same task in the outer loop (highlighted in orange), and then the loss will be used to update the model parameters.



### Algorithm 1 Model-Agnostic Meta-Learning

**Require:**  $p(\mathcal{T})$ : distribution over tasks

**Require:**  $\alpha, \beta$ : step size hyperparameters

```
1: randomly initialize  $\theta$ 
2: while not done do
3:   Sample batch of tasks  $\mathcal{T}_i \sim p(\mathcal{T})$ 
4:   for all  $\mathcal{T}_i$  do
5:     Evaluate  $\nabla_{\theta} \mathcal{L}_{\mathcal{T}_i}(f_{\theta})$  with respect to  $K$  examples
6:     Compute adapted parameters with gradient descent:  $\theta'_i = \theta - \alpha \nabla_{\theta} \mathcal{L}_{\mathcal{T}_i}(f_{\theta})$ 
7:   end for
8:   Update  $\theta \leftarrow \theta - \beta \nabla_{\theta} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i})$ 
9: end while
```

Figure 2. Model-Agnostic Meta-Learning. (Image from the [MAML paper](#))

MAML shows that learning could be more general and more flexible, which aligns with the direction of bypassing finetuning on each individual task. In the follow figure the authors of GPT-3 explained how this idea can be extended into learning language models when combined with in-context learning, with the outer loop iterates through different tasks, while the inner loop is described using **in-context learning**, which will be explained in more detail in later sections.

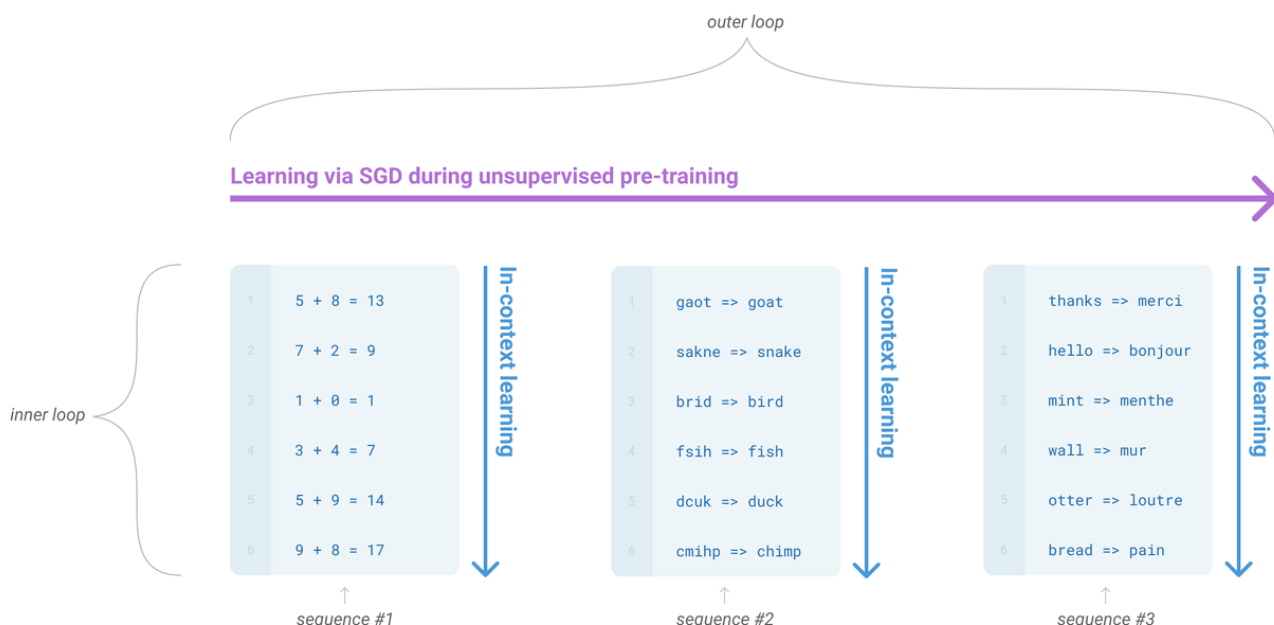


Figure 3. Language model meta-learning. (Image from [GPT-3 paper](#))

## *The Scale Hypothesis*

As perhaps the most influential idea behind the development of GPT-2 and GPT-3, the scale hypothesis refers to the observations that when training with larger data, large models could somehow develop new capabilities automatically without explicit supervision, or in other words, **emergent** abilities could occur when scaling up, just as what we saw in the zero-shot abilities of the pre-trained GPT-1.

Both GPT-2 and GPT-3 can be considered as experiments to test this hypothesis, with GPT-2 set to test whether a larger model pre-trained on a larger dataset could be directly used to solve down-stream tasks, and GPT-3 set to test whether in-context learning could bring improvements over GPT-2 when further scaled up.

We will discuss more details on how they implemented this idea in later sections.

## *In-Context Learning*

As we show in Figure 3, under the context of language models, in-context learning refers to the inner loop of the meta-learning process, where the model is given a natural language instruction and a few demonstrations of the task at inference time, and is then expected to complete that task by automatically discovering the patterns in the given demonstrations.

Note that in-context learning happens in the testing phase **with no gradient updates performed**, which is completely different from traditional finetuning and is more similar to how humans perform new tasks.

In case you are not familiar with the terminology, **demonstrations** usually means exemplary input-output pairs associated with a particular task, as we show in the "examples" part in the figure below:

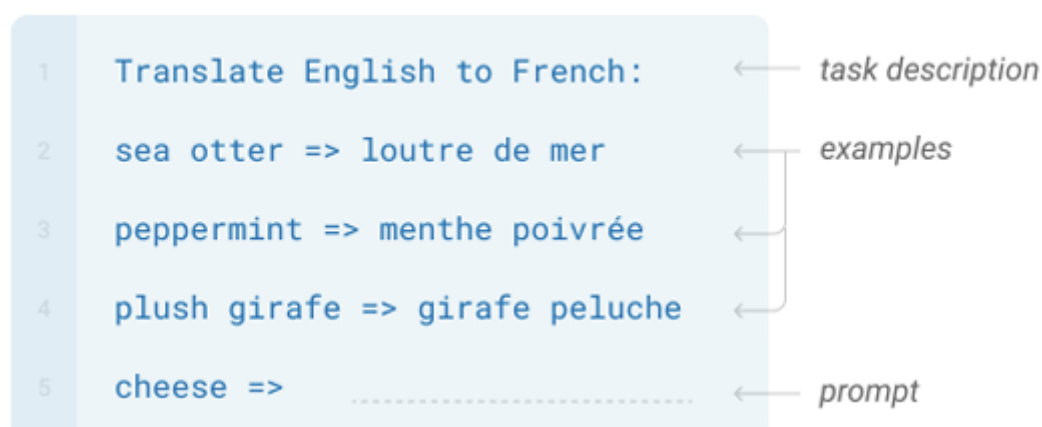


Figure 4. Example of few-shot in-context learning. (Image from [GPT-3 paper](#))

The idea of in-context learning was explored implicitly in GPT-2 and then more formally in GPT-3, where the authors defined three different settings: zero-shot, one-shot, and few-shot, depending on how many demonstrations are given to the model.

## The three settings we explore for in-context learning

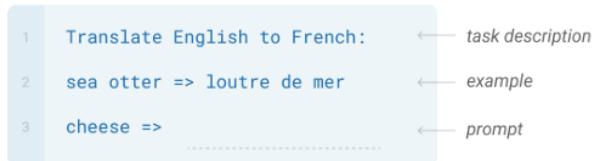
### Zero-shot

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



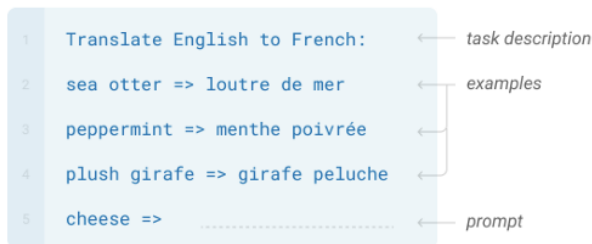
### One-shot

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 5. zero-shot, one-shot and few-shot in-context learning, contrasted with traditional finetuning. (Image from [GPT-3 paper](#))

In short, **task-agnostic learning highlights the potential of bypassing finetuning, while the scale hypothesis and in-context learning suggest a practical path to achieve that.**

In the following sections, we will walk through more details for GPT-2 and GPT-3, respectively.

## GPT-2

### *Model Architecture*

The GPT-2 model architecture is largely designed following GPT-1, with a few modifications:

- Moving LayerNorm to the input of each sub-block and adding an additional LayerNorm after the final self-attention block to make the training more stable.
- Scaling the weights of the residual layers by a factor of  $1/\sqrt{N}$ , where  $N$  is the number of residual layers.
- Expanding the vocabulary to 50257, and also using a modified BPE vocabulary.
- Increasing context size from 512 to 1024 tokens and using a larger batch size of 512.

In the GPT-2 paper, the authors trained four models with approximately log-uniformly spaced sizes, with number of parameter ranging from 117M to 1.5B:

Parameters	Layers	$d_{model}$
117M	12	768
345M	24	1024
762M	36	1280
1542M	48	1600

Table 1. Architecture hyperparameters for 4 GPT-2 models. (Image from [GPT-2 paper](#))

## Training Data

As we scale up the model we also need to use a larger dataset for training, and that is why in GPT-2 the authors created a new dataset called WebText, which contains about 45M links and is much larger than that used in pre-training GPT-1. They also mentioned lots of techniques to cleanup the data to improve its quality.

## Evaluation Results

Overall, GPT-2 achieved good results on many tasks, especially for language modeling related ones. However, for tasks like reading comprehension, translation and QA, it still performed worse than the respective SOTA models, which partly motivates the development of GPT-3.

Language Models are Unsupervised Multitask Learners										
	LAMBADA (PPL)	LAMBADA (ACC)	CBT-CN (ACC)	CBT-NE (ACC)	WikiText2 (PPL)	PTB (PPL)	enwik8 (BPB)	text8 (BPC)	WikiText103 (PPL)	1BW (PPL)
SOTA	99.8	59.23	85.7	82.3	39.14	46.54	0.99	1.08	18.3	<b>21.8</b>
117M	<b>35.13</b>	45.99	<b>87.65</b>	<b>83.4</b>	<b>29.41</b>	65.85	1.16	1.17	37.50	75.20
345M	<b>15.60</b>	55.48	<b>92.35</b>	<b>87.1</b>	<b>22.76</b>	47.33	1.01	<b>1.06</b>	26.37	55.72
762M	<b>10.87</b>	<b>60.12</b>	<b>93.45</b>	<b>88.0</b>	<b>19.93</b>	<b>40.31</b>	<b>0.97</b>	<b>1.02</b>	22.05	44.575
1542M	<b>8.63</b>	<b>63.24</b>	<b>93.30</b>	<b>89.05</b>	<b>18.34</b>	<b>35.76</b>	<b>0.93</b>	<b>0.98</b>	<b>17.48</b>	42.16

Table 2. GPT-2 zero-shot performance. (Image from [GPT-2 paper](#))

## GPT-3

### Model Architecture

GPT-3 adopted a very similar model architecture to that of GPT-2, and the only difference is that GPT-3 used an alternating dense and locally banded sparse attention patterns in Transformer.

GPT-3 trained 8 models with different sizes, with number of parameters ranging from 125M to 175B:

Model Name	$n_{\text{params}}$	$n_{\text{layers}}$	$d_{\text{model}}$	$n_{\text{heads}}$	$d_{\text{head}}$	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	$6.0 \times 10^{-4}$
GPT-3 Medium	350M	24	1024	16	64	0.5M	$3.0 \times 10^{-4}$
GPT-3 Large	760M	24	1536	16	96	0.5M	$2.5 \times 10^{-4}$
GPT-3 XL	1.3B	24	2048	24	128	1M	$2.0 \times 10^{-4}$
GPT-3 2.7B	2.7B	32	2560	32	80	1M	$1.6 \times 10^{-4}$
GPT-3 6.7B	6.7B	32	4096	32	128	2M	$1.2 \times 10^{-4}$
GPT-3 13B	13.0B	40	5140	40	128	2M	$1.0 \times 10^{-4}$
GPT-3 175B or “GPT-3”	175.0B	96	12288	96	128	3.2M	$0.6 \times 10^{-4}$

Table 3. Architecture hyperparameters for 8 GPT-3 models. (Image from [GPT-3 paper](#))

## Training Data

GPT-3 model was trained on even larger datasets, as listed in the table below, and again the authors did some cleanup work to improve data quality. Meanwhile, training datasets were not sampled in proportion to their size, but rather according to their quality, with high-quality dataset sampled more frequently during training.

Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

Table 4. Datasets used in GPT-3 training. (Image from [GPT-3 paper](#))

## Evaluation Results

By combining larger model with in-context learning, GPT-3 achieved strong performance on many NLP datasets including translation, question-answering, cloze tasks, as well as tasks require on-the-fly reasoning or domain adaptation. The authors presented very detailed evaluation results in the [original paper](#).

A few findings that we want to highlight in this article:

Firstly, during training of GPT-3 they observed a smooth scaling trend of performance with compute, as shown in the figure below, where the validation loss decreases linearly as compute increasing exponentially.



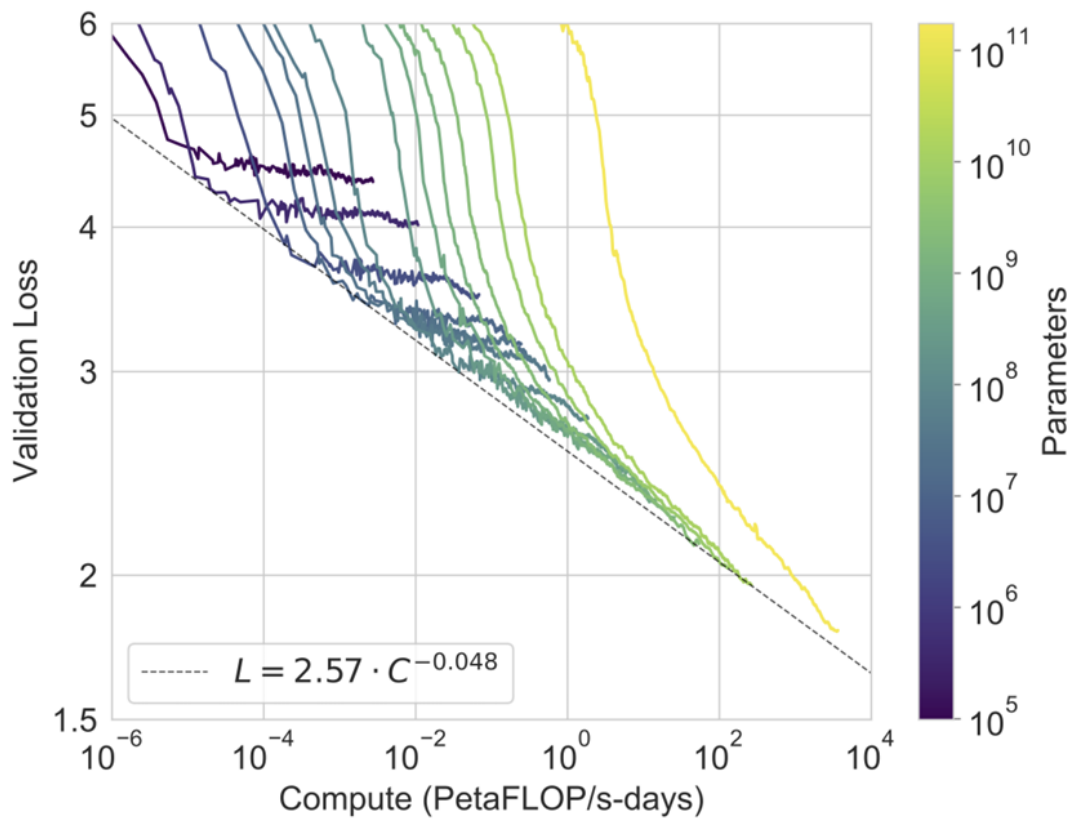


Figure 6. Smooth scaling of performance with compute. (Image from [GPT-3 paper](#))

Secondly, when comparing the three in-context learning settings (zero-shot, one-shot and few-shot), they observed that larger models appeared more efficient in all the three settings:

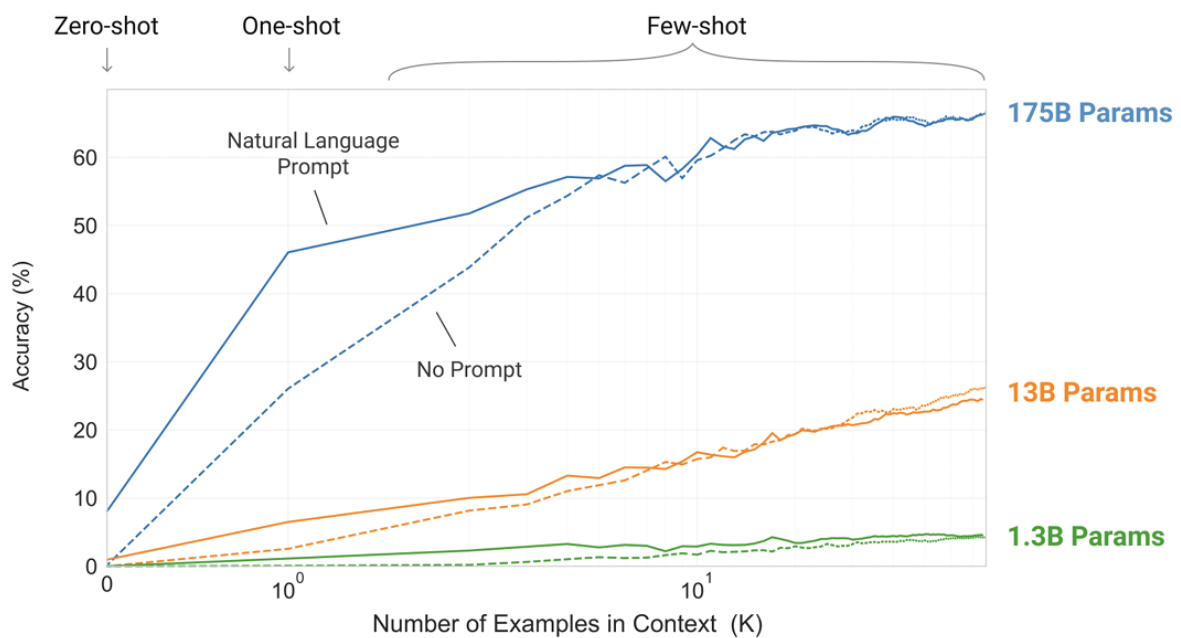


Figure 7. Larger models are more efficient in in-context learning. (Image from [GPT-3 paper](#))

Following that, they plotted the aggregate performance for all the three settings, which further demonstrated that larger models are more effective, and few-shot performance increased more rapidly than the other two settings.

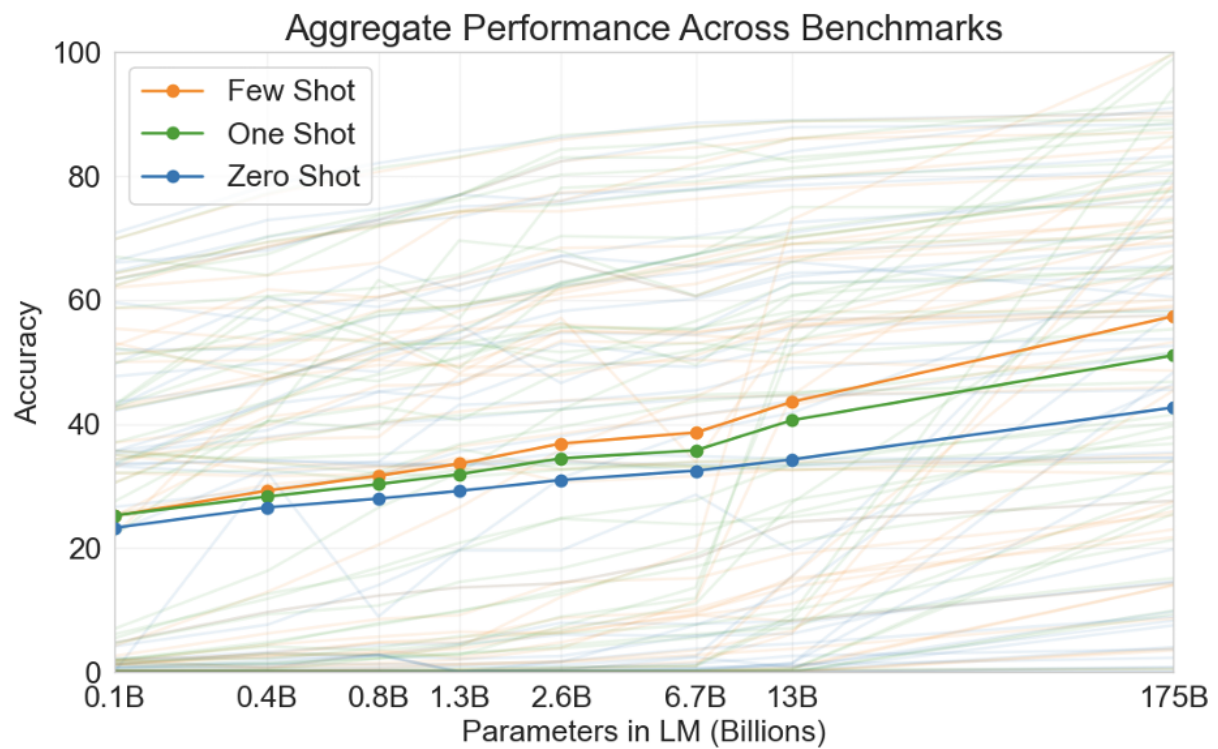


Figure 8. Aggregate performance for all 42 accuracy-denominated benchmarks. (Image from [GPT-3 paper](#))

---

## Conclusions

The development of GPT-2 and GPT-3 bridges the gap between the original GPT-1 with more advanced versions like InstructGPT, reflecting the ongoing refinement of OpenAI's methodology in training useful LLMs.

Their success also paves the way for new research directions in both NLP and the broader ML community, with many subsequent works focusing on understanding emergent capabilities, developing new training paradigms, exploring more effective data cleaning strategies, and proposing effective evaluation protocols for aspects like safety, fairness, and ethical considerations, etc.

In the next article, we will continue our exploration and walk you through the key elements of GPT-3.5 and InstructGPT.

Thanks for reading!

---