

# **Applied Machine Learning for Business Analytics**

Lecture 4: LLM I

# How to read model-card from Huggingface

Model distilled from DeepSeek-R1 based on Qwen

The screenshot shows a Huggingface model card page for a specific model version. At the top, the model name is displayed as "DeepSeek-R1-Distill-Qwen-32B-4bit". The "32B" and "4bit" parts are highlighted with red boxes. Two arrows point from these red boxes to the right: one arrow points upwards to the text "Model size", and another arrow points diagonally upwards and to the right to the text "Quantization Precision". Below the model name, there are several tags: "Text Generation", "Transformers", "Safetensors", "MLX", "qwen2", "conversational", and a partially visible tag starting with "t". At the bottom of the card, there are links for "Model card", "Files and versions", and "Community".

Model distilled from DeepSeek-R1 based on Qwen

mlx-community/DeepSeek-R1-Distill-Qwen-32B-4bit

Model size

Quantization Precision

Text Generation Transformers Safetensors MLX qwen2 conversational t

Model card Files and versions Community 1

# Agenda

1. Intro to LLM
2. GPT
3. Scaling Law
4. LLMs' Training

# 1. Intro to LLM

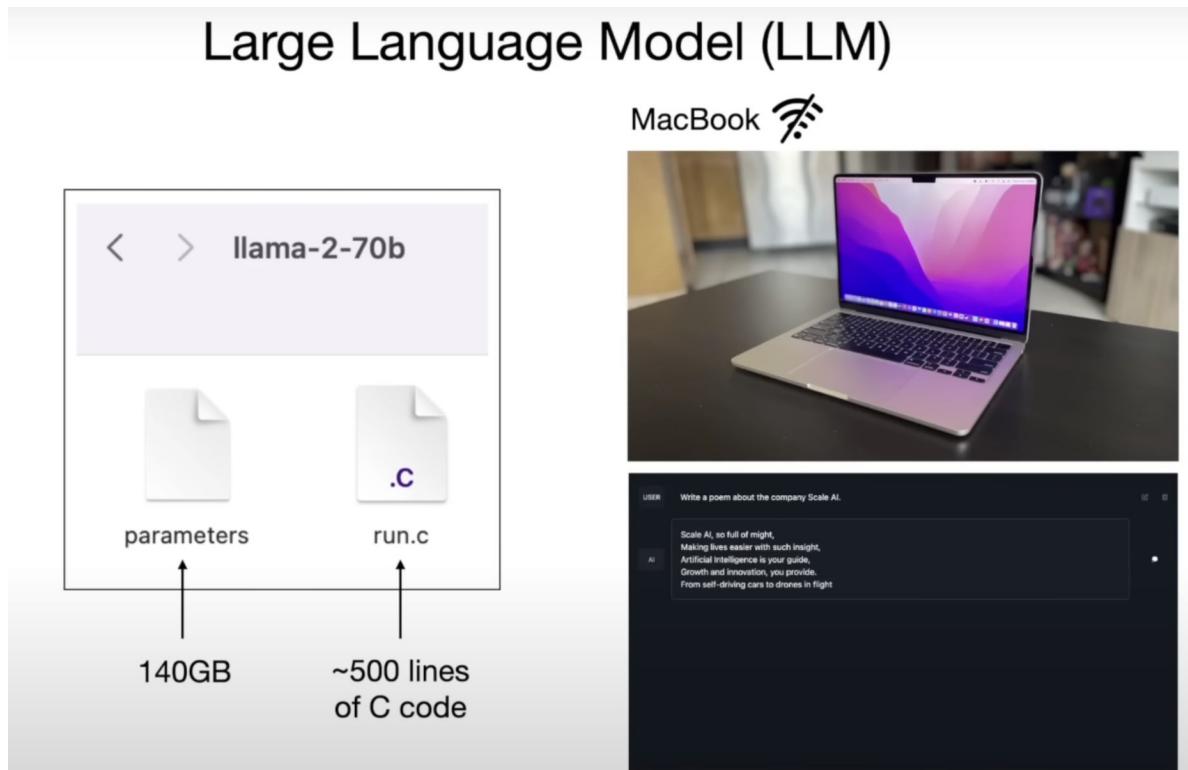
# What is LLM

Large language models (LLMs) mainly refer to transformer-based neural language models<sup>1</sup> that contain tens to hundreds of billions of parameters, which are pre-trained on massive text data, such as PaLM [31], LLaMA [32], and GPT-4 [33], as summarized in Table III. Compared

Source: <https://arxiv.org/pdf/2402.06196.pdf>

# What is LLM

## Large Language Model (LLM)



LLM  
Inference

Source:

[https://www.youtube.com/watch?v=zjkBMFhNj\\_q](https://www.youtube.com/watch?v=zjkBMFhNj_q)

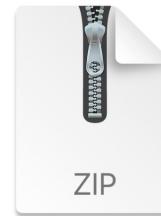
# LLM is compressing the Internet



## Language Modeling Is Compression

Grégoire Delétang<sup>\*1</sup>, Anian Ruoss<sup>\*1</sup>, Paul-Ambroise Duquenne<sup>2</sup>, Elliot Catt<sup>1</sup>, Tim Genewein<sup>1</sup>, Christopher Mattern<sup>1</sup>, Jordi Grau-Moya<sup>1</sup>, Li Kevin Wenliang<sup>1</sup>, Matthew Aitchison<sup>1</sup>, Laurent Orseau<sup>1</sup>, Marcus Hutter<sup>1</sup> and Joel Veness<sup>1</sup>

<sup>\*</sup>Equal contributions, <sup>1</sup>Google DeepMind, <sup>2</sup>Meta AI & Inria



Internet data  
~45TB of text

1024 A100 GPUs, 34 days  
\$5Mdata

GPT3 ~175B

# LLM's Evolution Process

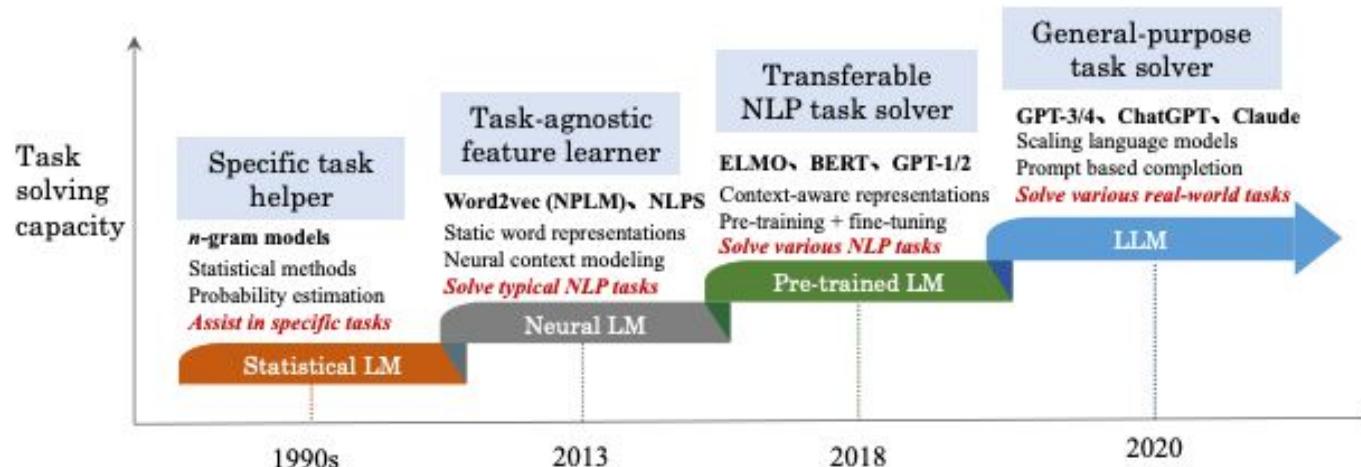
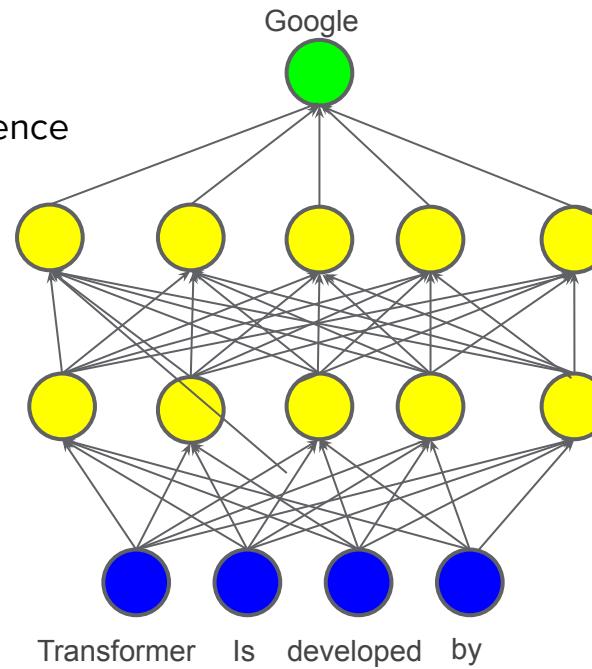


Fig. 2: An evolution process of the four generations of language models (LM) from the perspective of task solving capacity. Note that the time period for each stage may not be very accurate, and we set the time mainly according to the publish date of the most representative studies at each stage. For neural language models, we abbreviate the paper titles of two representative studies to name the two approaches: NPLM [1] ("A neural probabilistic language model") and NLPS [2] ("Natural language processing (almost) from scratch"). Due to the space limitation, we don't list all representative studies in this figure.

source: <https://arxiv.org/pdf/2303.18223.pdf>

# LLM Pretraining

- LLM
  - Pre-trained by **large-scale** unannotated corpus
- Training target: token prediction
  - For GPT/decoder: next token prediction in sequence
  - For BERT/encoder: in-context token prediction



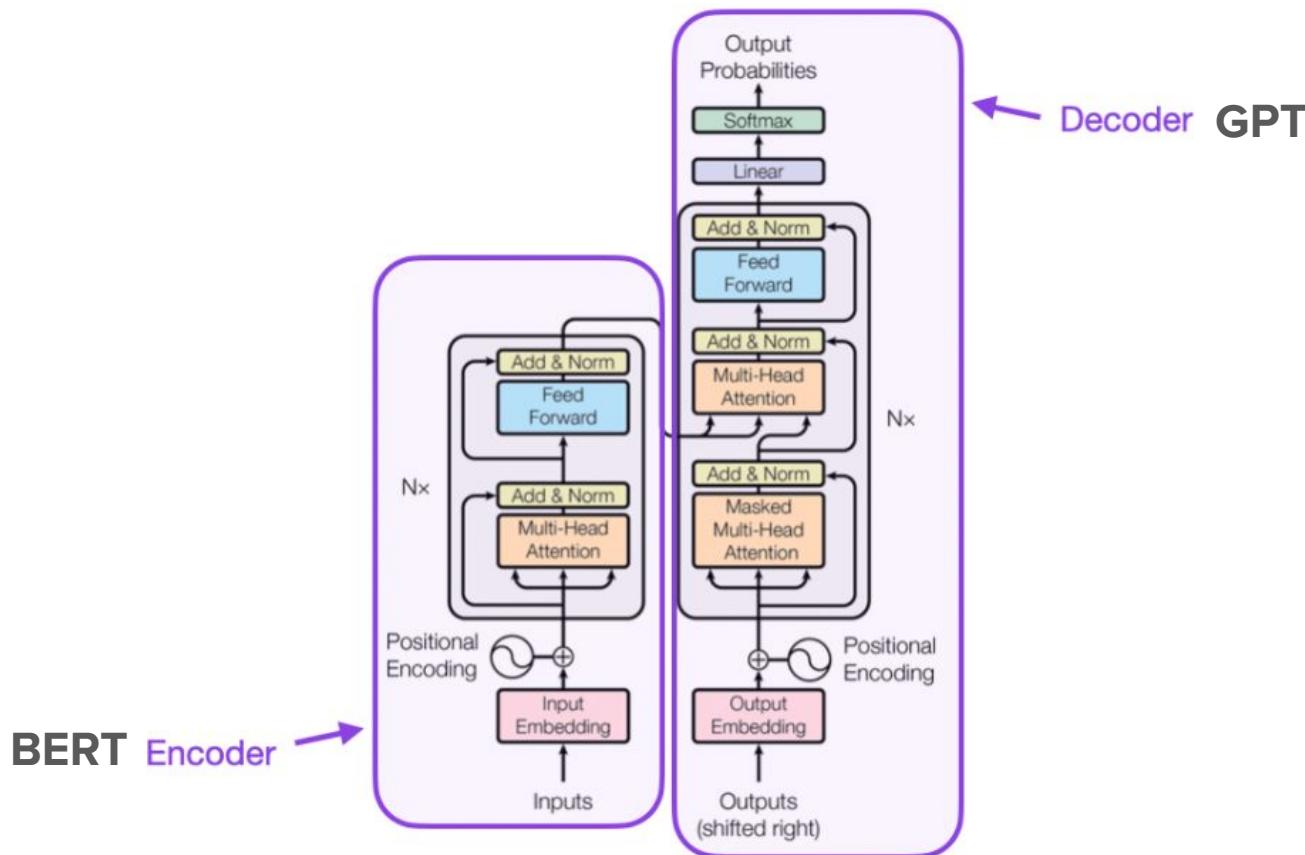
# Word prediction is absorbing knowledge

*"Transformer architecture" redirects here. For the design of electrical transformers, see [Transformer](#).*

A **transformer** is a [deep learning](#) architecture based on the [multi-head attention mechanism](#)<sup>[1]</sup>. It is notable for not containing any recurrent units, and thus requires less training time than previous recurrent neural architectures, such as [long short-term memory](#) (LSTM),<sup>[2]</sup> and its later variation has been prevalently adopted for training [large language models](#) on large (language) datasets, such as the [Wikipedia corpus](#) and [Common Crawl](#).<sup>[3]</sup> Input text is split into [n-grams](#) encoded as [tokens](#) and each token is converted into a vector via looking up from a word embedding table. At each layer, each token is then contextualized within the scope of the context window with other (unmasked) tokens via a parallel multi-head [attention mechanism](#) allowing the signal for key tokens to be amplified and less important tokens to be diminished. Though the transformer paper was published in 2017, the softmax-based attention mechanism was proposed in 2014 for [machine translation](#),<sup>[4][5]</sup> and the Fast Weight Controller, similar to a transformer, was proposed in 1992.<sup>[6][7][8]</sup>

This architecture is now used not only in [natural language processing](#) and [computer vision](#),<sup>[9]</sup> but also in [audio](#)<sup>[10]</sup> and multi-modal processing. It has also led to the development of [pre-trained systems](#), such as [generative pre-trained transformers](#) (GPTs)<sup>[11]</sup> and [BERT](#)<sup>[12]</sup> (Bidirectional Encoder Representations from Transformers).

# Transformer is the backbone of LLMs

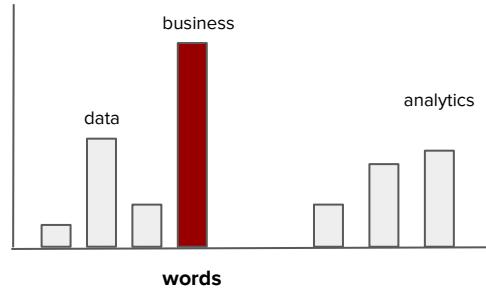


# How does it work

What is business analytics?

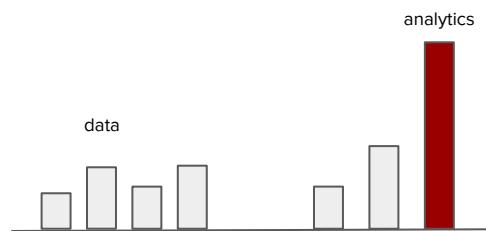


$\text{prob(next token} \mid \text{input text})$



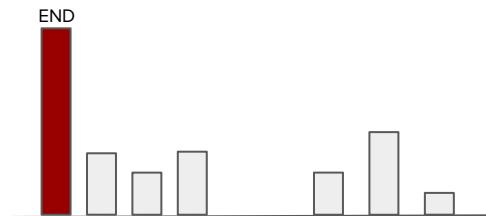
Business

What is business analytics? Business



analytics

What is business analytics? Business analytics is the practice.....



END

# How does it work

- It is a blackbox
  - Billions of parameters with nonlinear mapping
    - We can measure that this works
    - But we do not understand the full details that how those parameters collaborate to predict the next token
  - Sometimes, the performance is a bit strange and imperfect
    - Hallucination
    - Reversal Curse
    - Etc



How many 'm's are in the word 'Weather'?



There is one 'm' in the word 'Weather'.

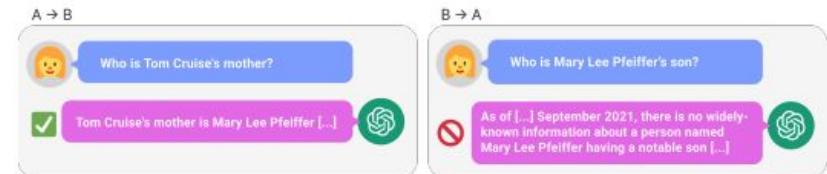
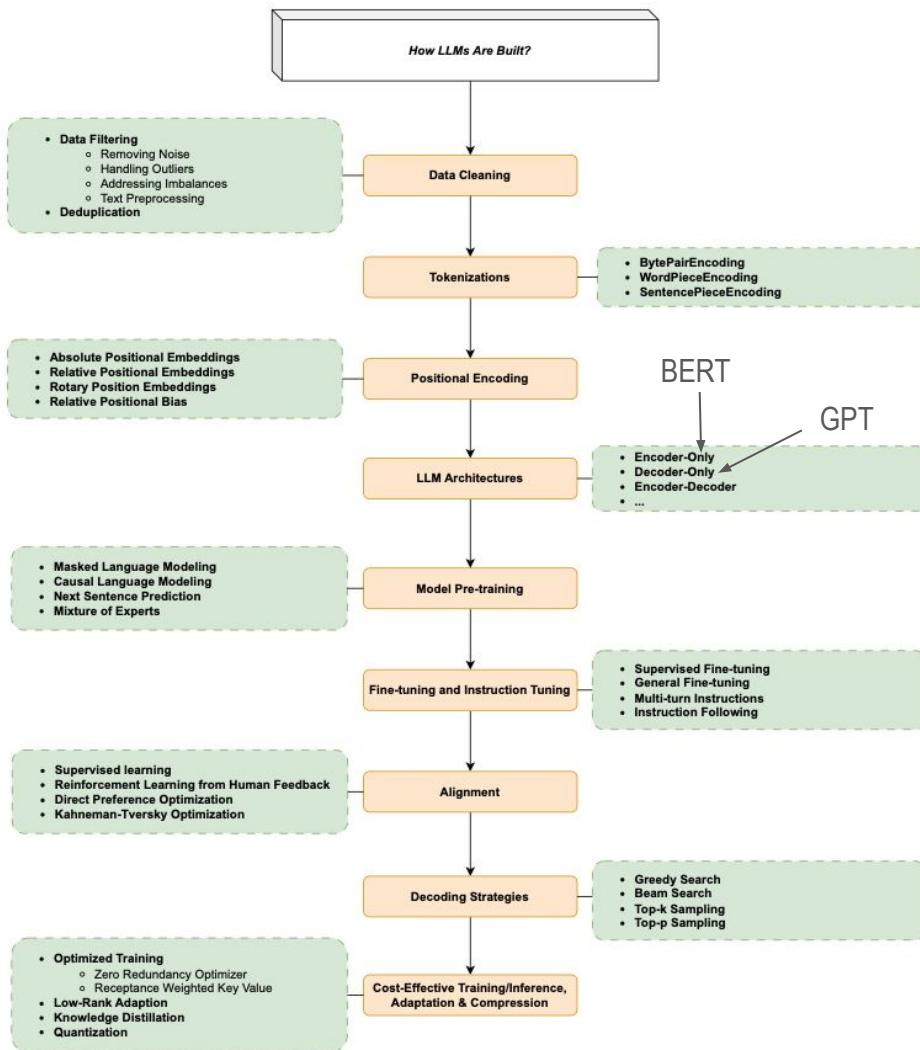


Figure 1: **Inconsistent knowledge in GPT-4.** GPT-4 correctly gives the name of Tom Cruise's mother (left). Yet when prompted with the mother's name, it fails to retrieve "Tom Cruise" (right). We hypothesize this ordering effect is due to the Reversal Curse. Models trained on "A is B" (e.g. "Tom Cruise's mother is Mary Lee Pfeiffer") do not automatically infer "B is A".

source : <https://arxiv.org/pdf/2309.12288.pdf>

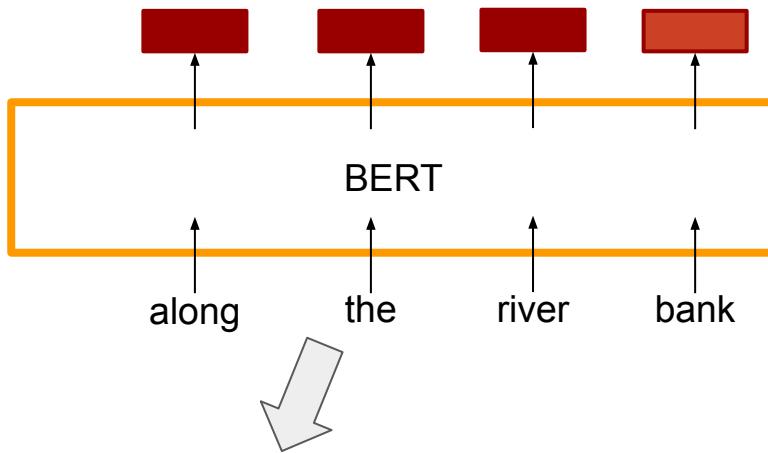


Source: <https://arxiv.org/pdf/2402.06196.pdf>

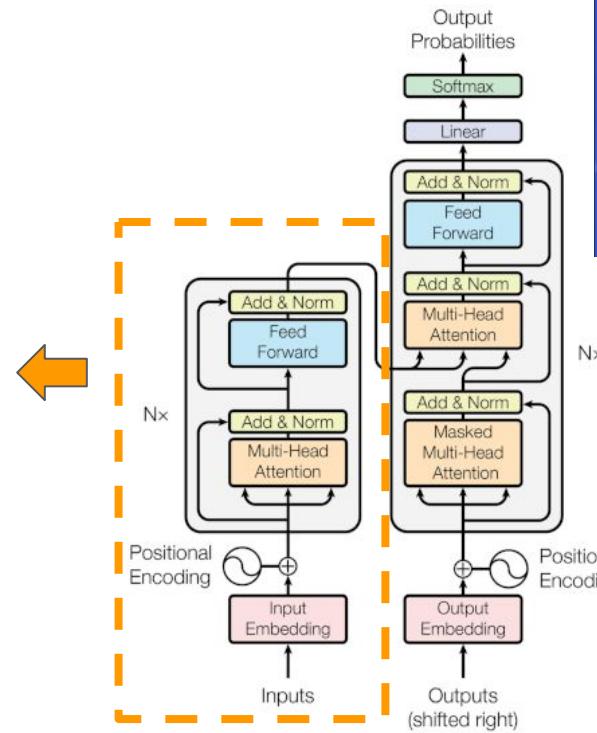
## **2. GPT**

# Recall on What is BERT

- Bidirectional Encoder Representations from Transformers (**BERT**)
- BERT: Encoder of Transformer,



Given a sequence of words, generate a sequence of vectors and then can be used for various NLP tasks



Transformer

Solve Seq2Seq Task

# What is GPT

- **Generative Pre-trained Transformer**
  - GPT: Decoder only of Transformer
  - Goal: Learn how to generate high-quality text

Language Models are Unsupervised Multitask Learners

## Improving Language Understanding by Generative Pre-Training

Alec Radford<sup>1,2</sup> Jeffrey Wu<sup>1,2</sup> Rewon Child<sup>1</sup> David Luan<sup>1</sup> Dario Amodei<sup>1,2</sup> Ilya Sutskever<sup>1</sup>

### Abstract

Natural language understanding poses a wide range of diverse tasks such as textual entailment, question answering, semantic similarity measurement, and document classification. While there is a large amount of labeled data available, labeled data for learning these specific tasks is scarce, making it challenging for downstream models to learn well. In this paper, we propose that instead of training a model on these tasks can be realized by generative pre-training of a language model on a large dataset of unlabeled text. We show that a generative pre-trained model can learn to perform well on these tasks without any explicit supervision when trained on a new dataset of unlabeled text. We demonstrate this by conditioning a document-level question, answer, and entailment model on the generated text from a 1.5B parameter OpenAI GPT pre-trained model. We also show that GPT can learn to perform well on a few-shot setting of a task by learning to generate examples demonstrating desired behavior for a desired task, train a system to imitate these behaviors, and then use the system to generate examples for the task. Finally, we show that GPT can learn to generate text in a few-shot setting without using the 127,000+ training examples. This has served well to make progress on narrow expert tasks such as reading comprehension and question answering. For example, on the CoQA dataset – matching or exceeding the performance of state-of-the-art systems (Yang et al., 2017), reading comprehension systems (Jia & Liang, 2017), and question answering systems (Rajpurkar et al., 2016). The capacity of the language model is essential to improving performance on these tasks. We find that increasing its size increases performance in a log-linear fashion. For example, our 1.5B parameter GPT achieves 85% accuracy on the SQuAD v1.1 dataset, while a 12B parameter GPT achieves 91% accuracy. Our general task agnostic model outperforms discriminatively trained models on the 12 tasks studied. For example, we achieve absolute improvements of 1.5% on textual entailment, 1.5% on question answering (RACIE), and 1.5% on textual entailment (MultiNLI).

1OpenAI, 2University of California Berkeley

### Abstract

Natural language processing tasks, such as question answering, machine translation, reading comprehension, and many others, are often approached with supervised learning on task-specific datasets. These datasets are often small, and the models begin to learn these tasks without any explicit supervision when trained on a new dataset of unlabeled text. In this paper, we propose that a generative pre-trained model can learn to perform well on these tasks without any explicit supervision when trained on a new dataset of unlabeled text. We demonstrate this by conditioning a document-level question, answer, and entailment model on the generated text from a 1.5B parameter OpenAI GPT pre-trained model. This has served well to make progress on narrow expert tasks such as reading comprehension and question answering. For example, on the CoQA dataset – matching or exceeding the performance of state-of-the-art systems (Yang et al., 2017), reading comprehension systems (Jia & Liang, 2017), and question answering systems (Rajpurkar et al., 2016). The capacity of the language model is essential to improving performance on these tasks. We find that increasing its size increases performance in a log-linear fashion. For example, our 1.5B parameter GPT achieves 85% accuracy on the SQuAD v1.1 dataset, while a 12B parameter GPT achieves 91% accuracy. Our general task agnostic model outperforms discriminatively trained models on the 12 tasks studied. For example, we achieve absolute improvements of 1.5% on textual entailment, 1.5% on question answering (RACIE), and 1.5% on textual entailment (MultiNLI).

## Language Models are Few-Shot Learners

Tom B. Brown <sup>1</sup>	Bojanja Name <sup>2</sup>	Nick Ryder <sup>2</sup>	Melanie Suhibai <sup>2</sup>
Jarou Kupka <sup>2</sup>	Pratik Dharwadkar <sup>2</sup>	Ariadna Subbiah <sup>2</sup>	Girish Savani <sup>2</sup>
Amanda Askell <sup>2</sup>	Sudhanshu Agarwal <sup>2</sup>	Ariadna Subbiah <sup>2</sup>	Ten Hoenigh <sup>2</sup>
Raven Child <sup>2</sup>	Abby Russek <sup>2</sup>	David M. Ziegler <sup>2</sup>	Jeffrey Wu <sup>2</sup>
Christopher House <sup>2</sup>	Mark Chen <sup>2</sup>	Eric Stigle <sup>2</sup>	Clement Winter <sup>2</sup>
Regina Catts <sup>2</sup>	Jack Clark <sup>2</sup>	Matthew Lewellen <sup>2</sup>	Scott Gray <sup>2</sup>
Sam McCandlish <sup>2</sup>	Alec Radford <sup>2</sup>	Ilya Sutskever <sup>2</sup>	Dario Amodei <sup>2</sup>

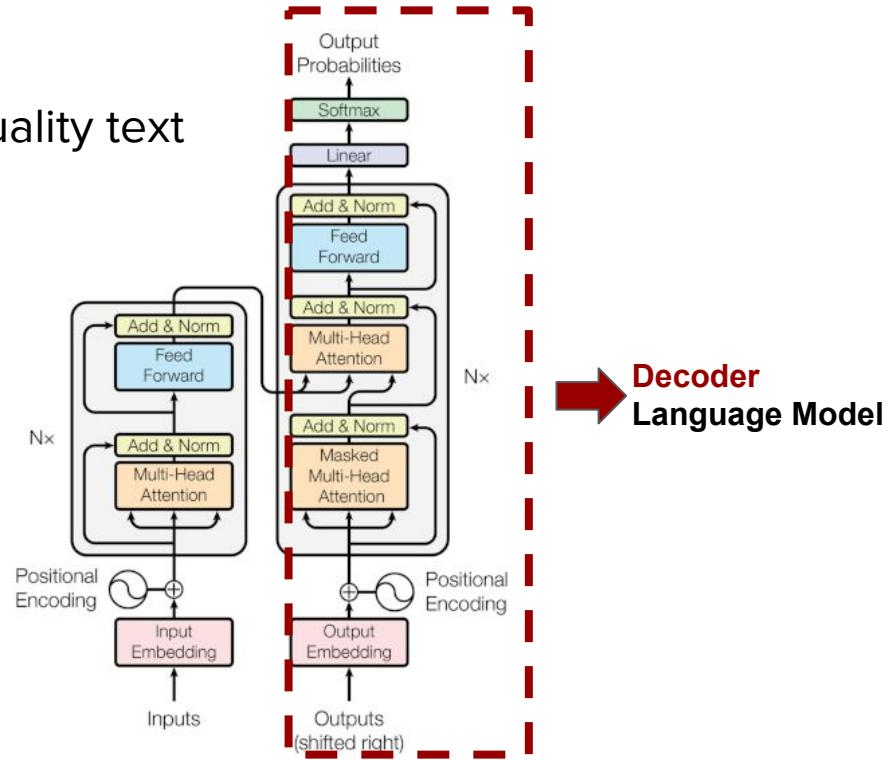
### Abstract

Recent work has demonstrated substantial gains in many NLP tasks and benchmarks by pre-training on a large corpus of text followed by fine-tuning on a specific task. While typically task-specific, this approach is often unable to learn to perform well on a new task with only a small number of examples. By contrast, humans can generally perform a new language task from only a few examples. In this paper, we show that GPT, a generative pre-trained model, can learn to perform well on a few-shot setting of a task. Specifically, we train GPT on 175 billion tokens of unlabeled text and then fine-tune it on a few-shot setting of a task. We show that GPT can learn to perform well on several NLP tasks with few-shot demonstrations, specifically purity via two interactions with the model. GPT3 can learn to generate text in a few-shot setting of a task, such as generating a response to a question and a clear task, as well as several tasks that require the GPT to reason or domain adapt, such as translating between domains, generating text in a new language, and generating text in a new style. At the same time, we also identify some datasets where GPT's few-shot learning still struggles, as well as some datasets where GPT's few-shot learning is better than its full-fine-tuned performance. Finally, we find that GPT3 can generate samples of news articles which human evaluators have difficulty distinguishing from written by humans. We discuss broader societal impacts of this finding and GPT3 in general.

GPT1

GPT2

GPT3



# GPT3

## Language Models are Few-Shot Learners

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

OpenAI

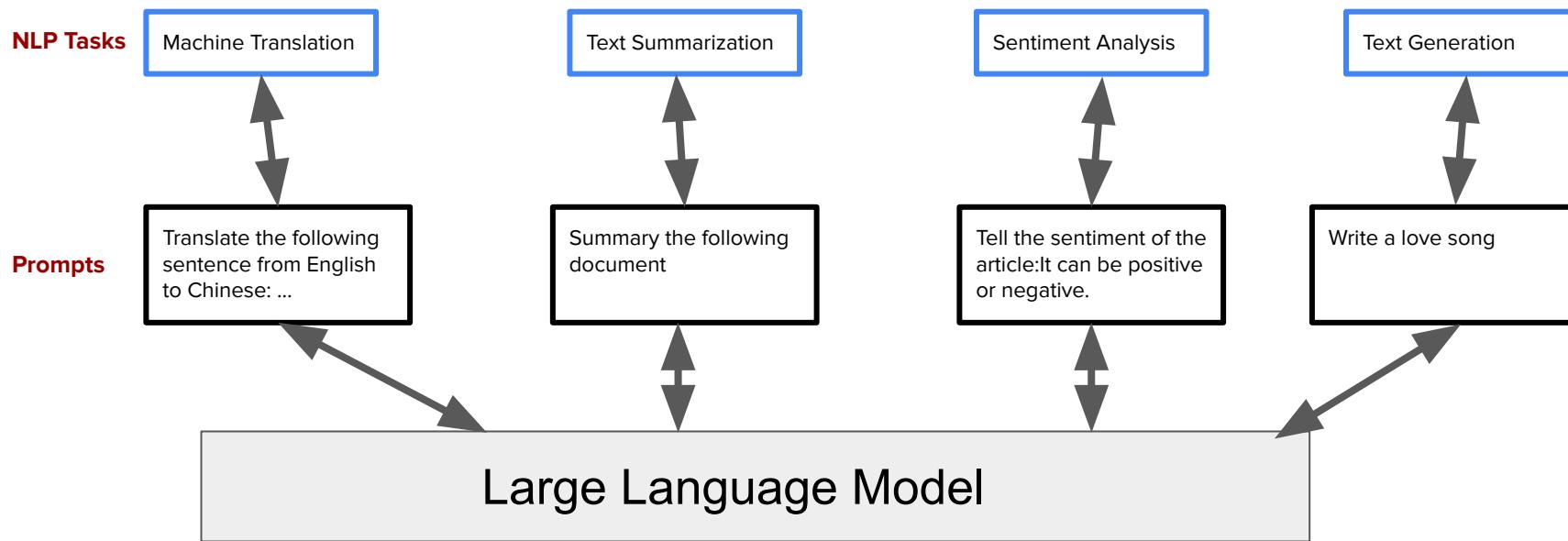
### Abstract

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

1. Compared to GPT2 and GPT1, more parameters and bigger training dataset are used in GPT3. And the generalization capability is named as **in-context learning**.
2. GPT2 has constraints in handling certain specific tasks while GPT3 show groundbreaking abilities. So it has paved the way for even bigger and more complex models

# In-context Learning

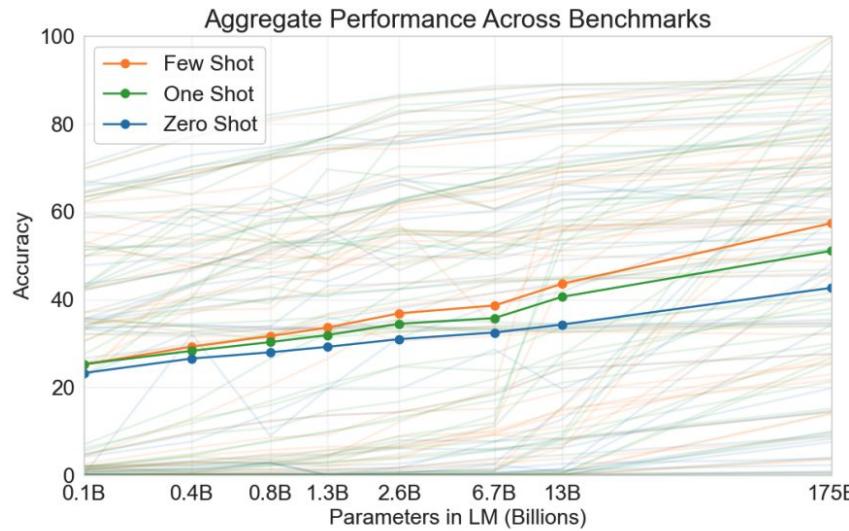
In-context learning: using the text input of a pre-trained language model as a form of task specification: the model is conditioned on a natural language instruction and/or a few demonstrations of the task and is then expected to complete further instances of the task simply by predicting what comes next.



# GPT3: Prompting

- Zero-shot Prompting
  - No examples are given in prompt
  - “Please answer,  $3+2=?$ ”
- One-shot Prompting
  - One example is given
  - “ $1+7=8$ , please answer,  $3+2=?$ ”
- Few-shot Prompting
  - A few shot examples of tasks are provided
  - “ $1+1=2$ ,  $1+7=8$ , please answer,  $3+2=?$ ”

# GPT3: Performances of Prompting



**Figure 1.3: Aggregate performance for all 42 accuracy-denominated benchmarks** While zero-shot performance improves steadily with model size, few-shot performance increases more rapidly, demonstrating that larger models are more proficient at in-context learning. See Figure 3.8 for a more detailed analysis on SuperGLUE, a standard NLP benchmark suite.

Source: <https://arxiv.org/pdf/2005.14165.pdf>

# BERT vs GPT

## BERT

- Architecture:
  - Transformer Encoder block
  - Less training parameters (a few hundred M)
- Model learning:
  - Two objectives: masked language model (cbow) and next sentence prediction
  - Bi-directional
  - Less training data
- Applications:
  - Traditional NLP Tasks: summarization, classification, representation learning, information retrieval

## GPT

- Architecture:
  - Transformer Decoder block
  - More training parameters (a few hundred B)
- Model learning:
  - Generative, next word prediction
  - Uni-directional (left to right)
  - More training data
- Applications:
  - Natural language generation, Q/A, chatbot

# BERT vs GPT: BERT was winning

**Bert:** Pre-training of deep bidirectional transformers for language understanding

J Devlin, MW Chang, K Lee, K Toutanova - arXiv preprint arXiv ..., 2018 - arxiv.org

... We introduce **BERT** and its detailed implementation in this ... For finetuning, the **BERT** model is first initialized with the pre ... A distinctive feature of **BERT** is its unified architecture across ...

☆ Save 99 Cite Cited by 82519 Related articles All 46 versions ☰

[PDF] Improving language understanding by generative pre-training

A Radford, K Narasimhan, T Salimans, I Sutskever

2018 · mikecaptain.com

## Abstract

Natural language understanding comprises a wide range of diverse tasks such as textual entailment, question answering, semantic similarity assessment, and document classification. Although large unlabeled text corpora are abundant, labeled data for learning these specific tasks is scarce, making it challenging for discriminatively trained models to perform adequately. We demonstrate that large gains on these tasks can be realized by generative pre-training of a language model on a diverse corpus of unlabeled

SHOW MORE ✕

☆ Save 99 Cite Cited by 7012 Related articles All 15 versions ☰

## Language models are few-shot learners

T Brown, B Mann, N Ryder... - Advances in neural ..., 2020 - proceedings.neurips.cc

... up **language models** greatly improves task-agnostic, **few-shot** ... GPT-3, an autoregressive **language model** with 175 billion ... **language model**, and test its performance in the **few-shot** ...

☆ Save 99 Cite Cited by 16390 Related articles All 27 versions ☰

At beginning, BERT got more adoption from NLP community compared to GPT and GPT2 (82k citation vs 23k citation)

# BERT vs GPT: Different Mindsets

## BERT

- Understand the language first before generating a response
- An encoder to learn the representation is the backbone
- Fine tune for specific tasks

## GPT

- Mainly focus on predicting the next token
- The decoder to predict the next token
- One shot or few-shot prompting without fine-tuning
- Scaling up parameters

# BERT vs GPT: GPT method is the SOTA now

- With the popularity of ChatGPT, GPT method is winning now
  - We understand others by the response
  - Representations or encodings does not matter, we can rely on outputs for any specific tasks
  - Closer to the idea of General AI (only one model)



About 216,000,000 results (0.34 seconds)

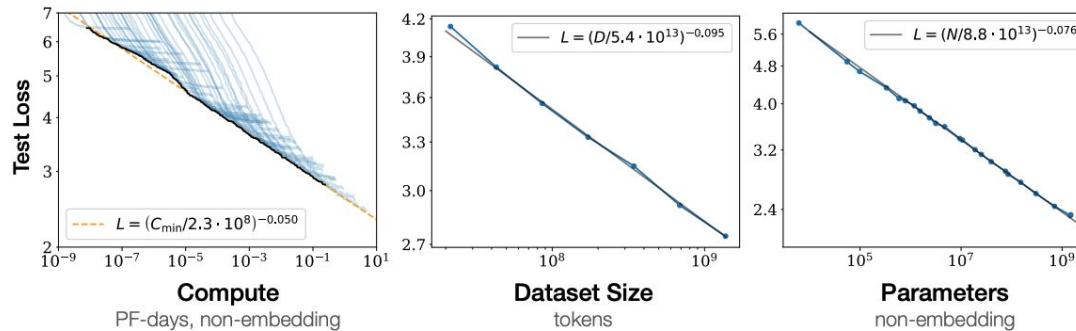


About 587,000,000 results (0.37 seconds)

### 3. Scaling laws

# What is “Large” in LLM

- The scale of LLM is defined in three aspects:
  - Model size
  - Dataset size
  - Amount of computing power for training

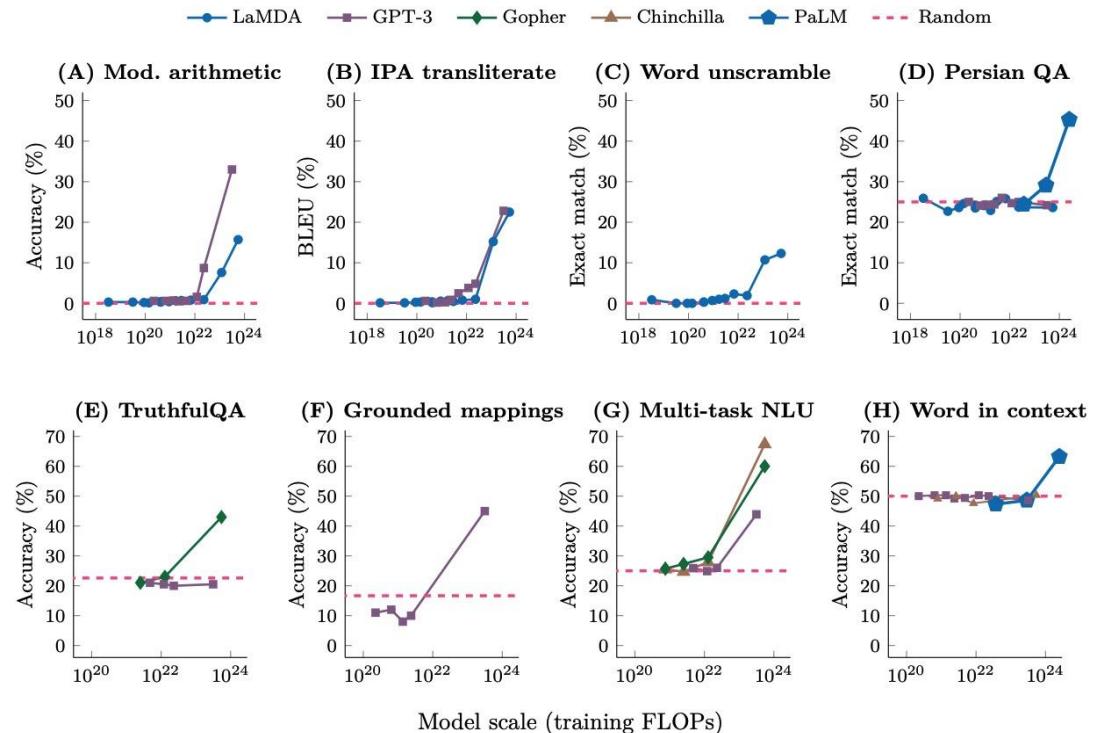


**Figure 1** Language modeling performance improves smoothly as we increase the model size, dataset size, and amount of compute<sup>2</sup> used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

Source: <https://arxiv.org/pdf/2001.08361.pdf>

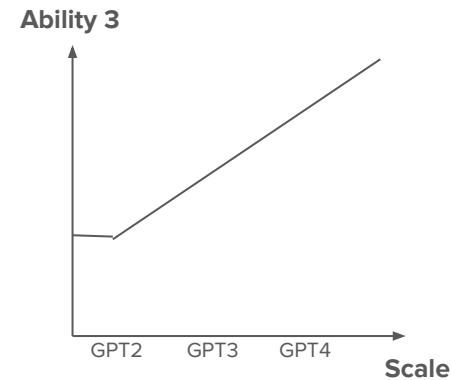
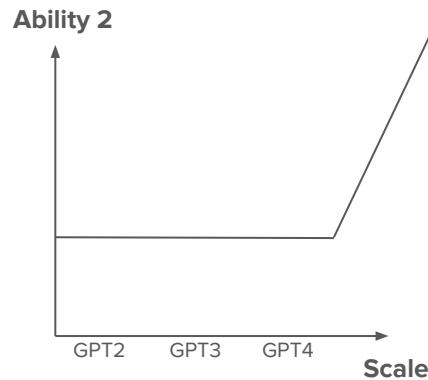
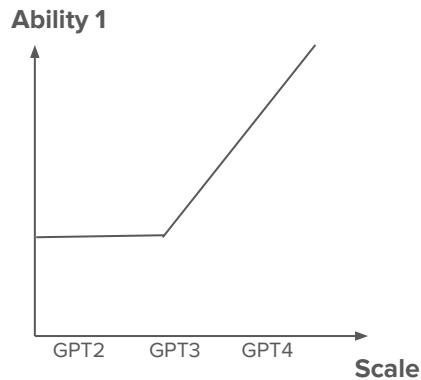
# Emergent abilities of large language models

Emergence is when quantitative changes in a system result in qualitative changes in behavior



Source: <https://arxiv.org/abs/2206.07682>

# Simplified View of Emergent Abilities



Therefore, some researchers think even some abilities do not work with the current LLMs, we should think once larger models are available, many problems can be solved.

# The Scale of GPTs and BERT

<i>Model Version</i>	<i>Architecture</i>	<i>Parameter count</i>	<i>Training data</i>
<b>Bert-base</b>	12-level, 12-headed Transformer encoder	0.11 billion	Toronto BookCorpus and English Wikipedia (3,200 million words)
<b>Bert-Large</b>	24-level, 16-headed Transformer encoder	0.34 billion	Toronto BookCorpus and English Wikipedia (3,200 million words)
<b>GPT1</b>	12-level, 12 headed Transformer decoder, followed by linear-softmax	0.12 billion	BookCorpus, 4.5 GB of text
<b>GPT2</b>	GPT-1, but with modified normalization	1.5 billion	WebText: 40 GB of text, 8 million documents
<b>GPT3</b>	GPT-2 but with modification to allow larger scaling	175 billion	570 GB plaintext, 0.4 trillion tokens

\* The estimated model size of GPT4 is around 175B to 280B.

# What is the scale of 175b

GPT-1  
BERT-Base



Levi 兵長 1.6m



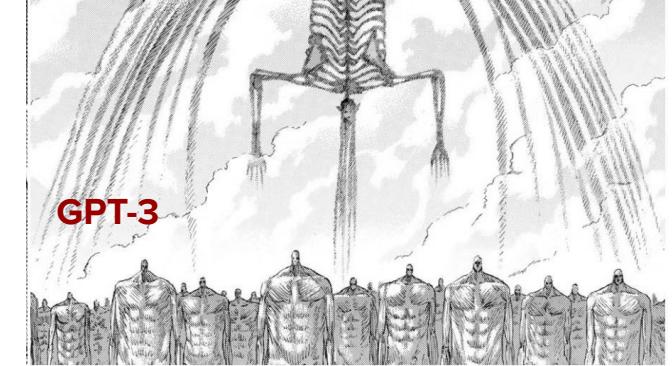
Cart Titan 車力の巨人 4m

BERT-Large

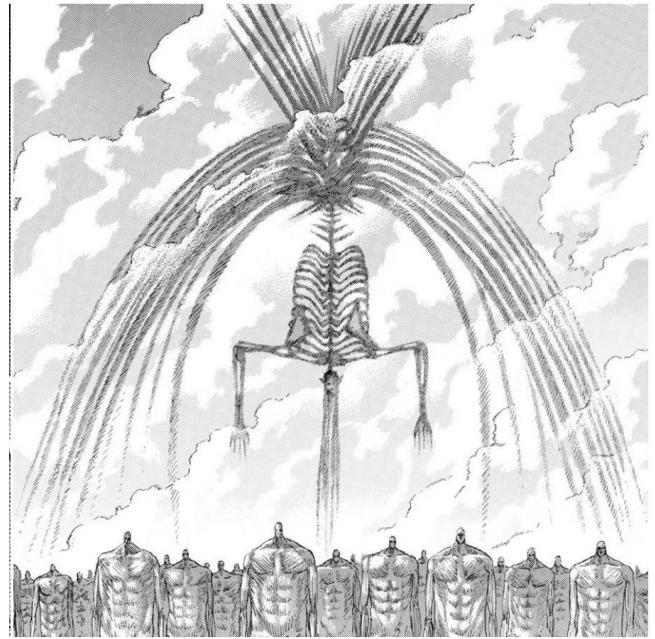


Beast Titan 獣の巨人 17m

GPT-2



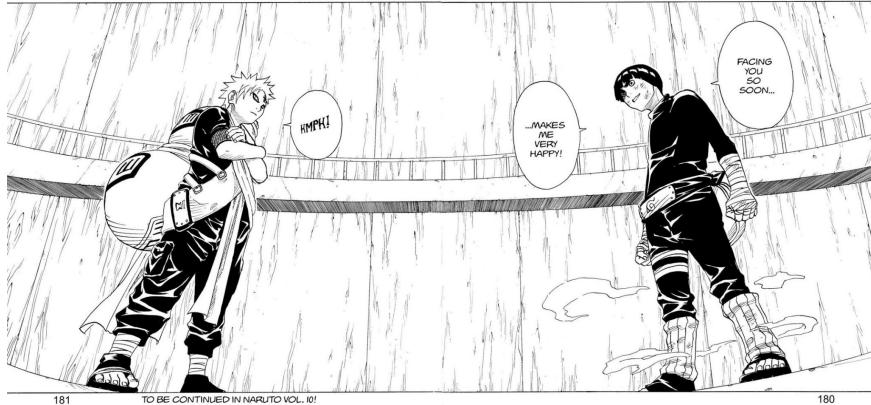
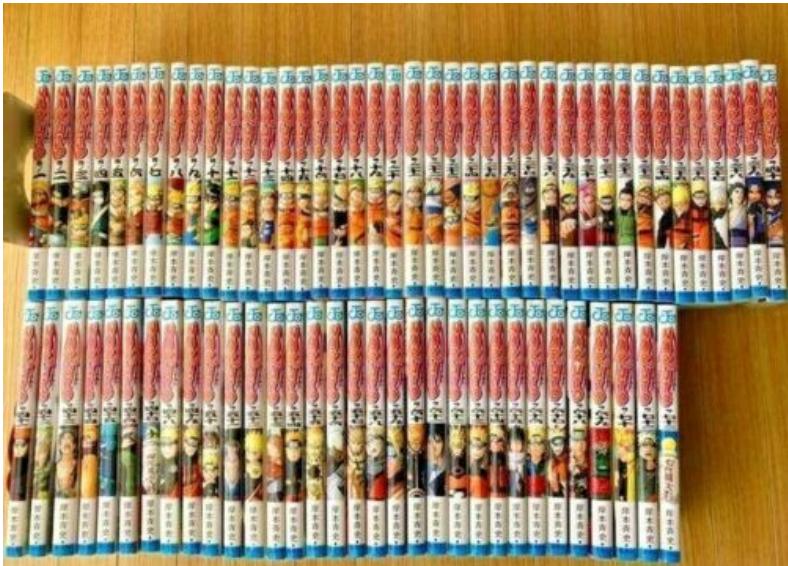
GPT-3



Two Stacked Eren Founding Titan 始祖の巨人  $2 \times 1000\text{m}$  <sup>31</sup>

# What is the scale 570GB

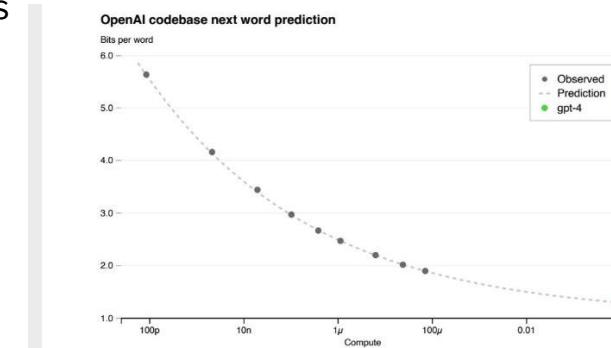
Read Naruto **270K** times



Only lines/text are counted.

# Predicted Performances of LLM

- Since training massive models requires significant investment, we need approaches to predict performance before committing resources
  - Justify AI high Capex for Big Tech & Startups companies
- Scaling laws provide a solution
  - Train multiple smaller models with different configurations
  - Derive scaling relationships from their performances
  - Extrapolate to predict large model performances



**Figure 1.** Performance of GPT-4 and smaller models. The metric is final loss on a dataset derived from our internal codebase. This is a convenient, large dataset of code tokens which is not contained in the training set. We chose to look at loss because it tends to be less noisy than other measures across different amounts of training compute. A power law fit to the smaller models (excluding GPT-4) is shown as the dotted line; this fit accurately predicts GPT-4's final loss. The x-axis is training compute normalized so that GPT-4 is 1.

# End of Scaling Law?



Pre-training as we know it will end

Compute is growing:

- Better hardware
- Better algorithms
- Larger clusters

Data is not growing:

- We have but one internet
- The fossil fuel of AI

Internet. We have, but one Internet. You could even say you can even go as far as to say. That data is the fossil fuel of AI. It was like, created somehow. And now we use it.



Sam Altman    
@sama

**there is no wall**

2:06 pm · 14 Nov 2024 · 2.5M Views

# Thoughts on the debate

- Computing power is growing rapidly while data growth is constrained by web scraping limitation and high-quality data is limited.
- Synthetic data could help but can not solve the problem
  - The quality might not be high
  - Most of the valuable human-created content would be used up
- What might be the solution?
  - Use more computing power to generate better synthetic data during testing
    - Train a model
    - Use scaled-up **inference compute** to generate rich synthetic data
    - Use that data to further improve the model through training

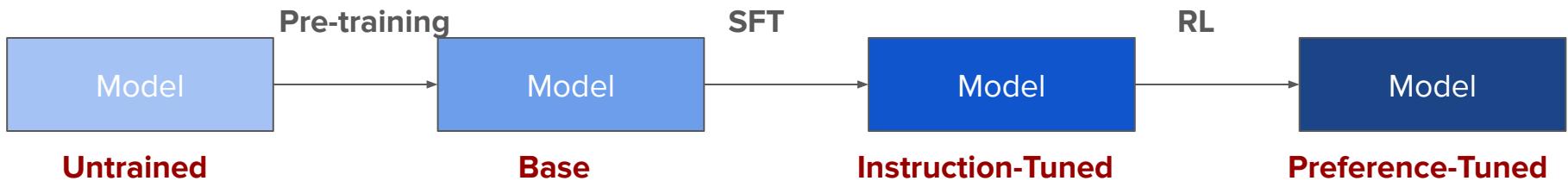
## 4. LLMs' Training

# Next Token Prediction

Training Objective:

Given a sequence of words  $X = (x_1, x_2, x_3, \dots, x_t)$ , predict the next token  $x_{t+1}$

# Three steps of training a high-quality LLM



# Pre-training

- The model is trained to predict the next word using a massive amount of web data.
- It results in a base model.
- Good:
  - “Cheap” without human annotations
  - Can absorb knowledge
- However, it is not good at following instructions or does not know our human's preference
  - The model needs **alignment**.
    - So it comes to instruction and preference tuning.

# Why do we need Alignment

What is the purpose of the list C in the code below?

```
def binomial_coefficient(n, r):
    C = [0 for i in range(r + 1)];
    C[0] = 1;
    for i in range(1, n + 1):
        j = min(i, r);
        while j > 0:
            C[j] += C[j - 1];
            j -= 1;
    return C[r]
```



**With alignment**

The list C in this code is used to store the values of the binomial coefficient as the function iterates through the values of n and r. It is used to calculate the value of the binomial coefficient for a given value of n and r, and stores the result in the final return value of the function.



What is the purpose of the list C in the code below?

```
def binomial_coefficient(n, r):
    C = [0 for i in range(r + 1)];
    C[0] = 1;
    for i in range(1, n + 1):
        j = min(i, r);
        while j > 0:
            C[j] += C[j - 1];
            j -= 1;
    return C[r]
```



**Without alignment**

source: <https://arxiv.org/pdf/2203.02155.pdf>

# Why do we need Alignment

What is the purpose of the list C in the code below?

```
def binomial_coefficient(n, r):
    C = [0 for i in range(r + 1)];
    C[0] = 1;
    for i in range(1, n + 1):
        j = min(i, r);
        while j > 0:
            C[j] += C[j - 1];
            j -= 1;
    return C[r]
```



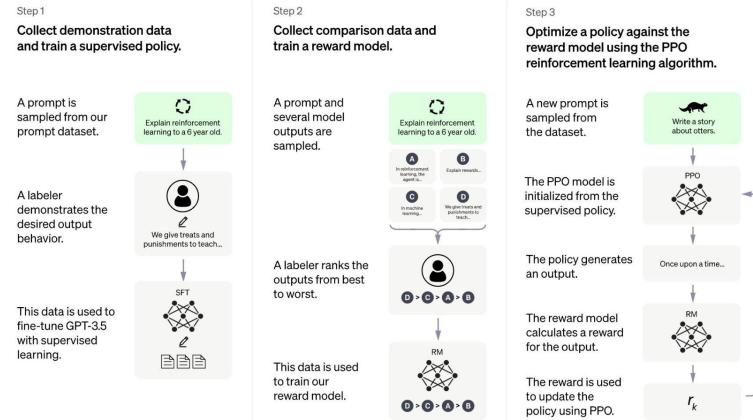
- A. to store the value of C[0]
- B. to store the value of C[1]
- C. to store the value of C[i]
- D. to store the value of C[i - 1]



source: <https://arxiv.org/pdf/2203.02155.pdf>

# Alignments make ChatGPT usable

- This is how ChatGPT is different from GPT3.
- Step 1: Instruction Tuning
- Step 2&3: Preference Tuning



## Training language models to follow instructions with human feedback

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

Pamela Mishkin\* Chong Zhang Sandhini Agarwal Katarina Slama Alex Ray

John Schulman Jacob Hilton Fraser Kelton Luke Miller Maddie Simens

Amanda Askell† Peter Welinder Paul Christiano†

Jan Leike\* Ryan Lowe\*

OpenAI

## Abstract

## Methods

We trained this model using Reinforcement Learning from Human Feedback (RLHF), using the same methods as InstructGPT, but with slight differences in the data collection setup. We trained an initial model using supervised fine-tuning: human AI trainers provided conversations in which they played both sides—the user and an AI assistant. We gave the trainers access to model-written suggestions to help them compose their responses. We mixed this new dialogue dataset with the InstructGPT dataset, which we transformed into a dialogue format.

source: <https://openai.com/blog/chatgpt>

ChatGPT should be quite close to InstructGPT in terms of implementation

# Alignment: SFT

- Supervised Fine-tuning: training data would be a pair of (prompt, responses)
  - For example,
    - Prompt: what is  $1+1$ ?
    - Response:  $1+1$  is equal to 2.
- With the above data, the model is forced to learn from demonstrated response to the prompts
- How to prepare those SFT data?
  - Let us check instructGPT

# Start with Prompts

- Prompts: query sent to GPT models
- Prompts are generated in the following ways:
  - Plain: ask the labelers to come up with an arbitrary task
  - Few-shot: ask the labelers to come up with an instruction
    - Like any coding/programming related questions
  - User-based: collected use-cases from OpenAI API users. And labelers are asked to come up with related prompts

Examples	Capabilities	Limitations
"Explain quantum computing in simple terms" →	Remembers what user said earlier in the conversation	May occasionally generate incorrect information
"Got any creative ideas for a 10 year old's birthday?" →	Allows user to provide follow-up corrections	May occasionally produce harmful instructions or biased content
"How do I make an HTTP request in Javascript?" →	Trained to decline inappropriate requests	Limited knowledge of world and events after 2021



# Response

- Prepare SFT dataset
  - It only has 13K training prompts with labeler demonstration
  - The ground-truth responses are provided directly
- Fine-tune the GPT model using the SFT dataset
  - Training target is the same as the pre-training: next word prediction (only on responses)

# Alignment is built upon human efforts



BUSINESS • TECHNOLOGY

## Exclusive: OpenAI Used Kenyan Workers on Less Than \$2 Per Hour to Make ChatGPT Less Toxic

15 MINUTE READ

source: <https://time.com/6247678/openai-chatgpt-kenya-workers/>

# SFT can not be scaled easily

- Let us look at a 7th grade problem as the prompt: What is  $x$  if

$$\sqrt{2x + 1} - 2 = x - 3.$$

- What is the correct answer?
- It would be very expensive to prepare the correct responses to prompts, especially those complex, creative prompts/instructions like math, reasoning problems.

# How about this?

What is  $x$  if  $\sqrt{2x + 1} - 2 = x - 3$ .

- A. 0
- B. 4
- C. 2
- D. 3

## **MCQ is easier**

**It is often much easier to compare Answers instead of writing Answers.**

# Preference Tuning

- RLHF: Reinforcement Learning from Human Feedback
    - The approach used for ChatGPT
    - It will start from the instruction-tuned model or base model (SFT can be skipped as DeepSeek R1-ZERO)
    - It has two steps:
      - Gather data and train a reward model
      - Fine-tune the LM with reinforcement learning
- 

## Deep Reinforcement Learning from Human Preferences

**Paul F Christiano**  
OpenAI  
[paul@openai.com](mailto:paul@openai.com)

**Jan Leike**  
DeepMind  
[leike@google.com](mailto:leike@google.com)

**Tom B Brown**  
Google Brain\*  
[tombrown@google.com](mailto:tombrown@google.com)

**Miljan Martic**  
DeepMind  
[miljanm@google.com](mailto:miljanm@google.com)

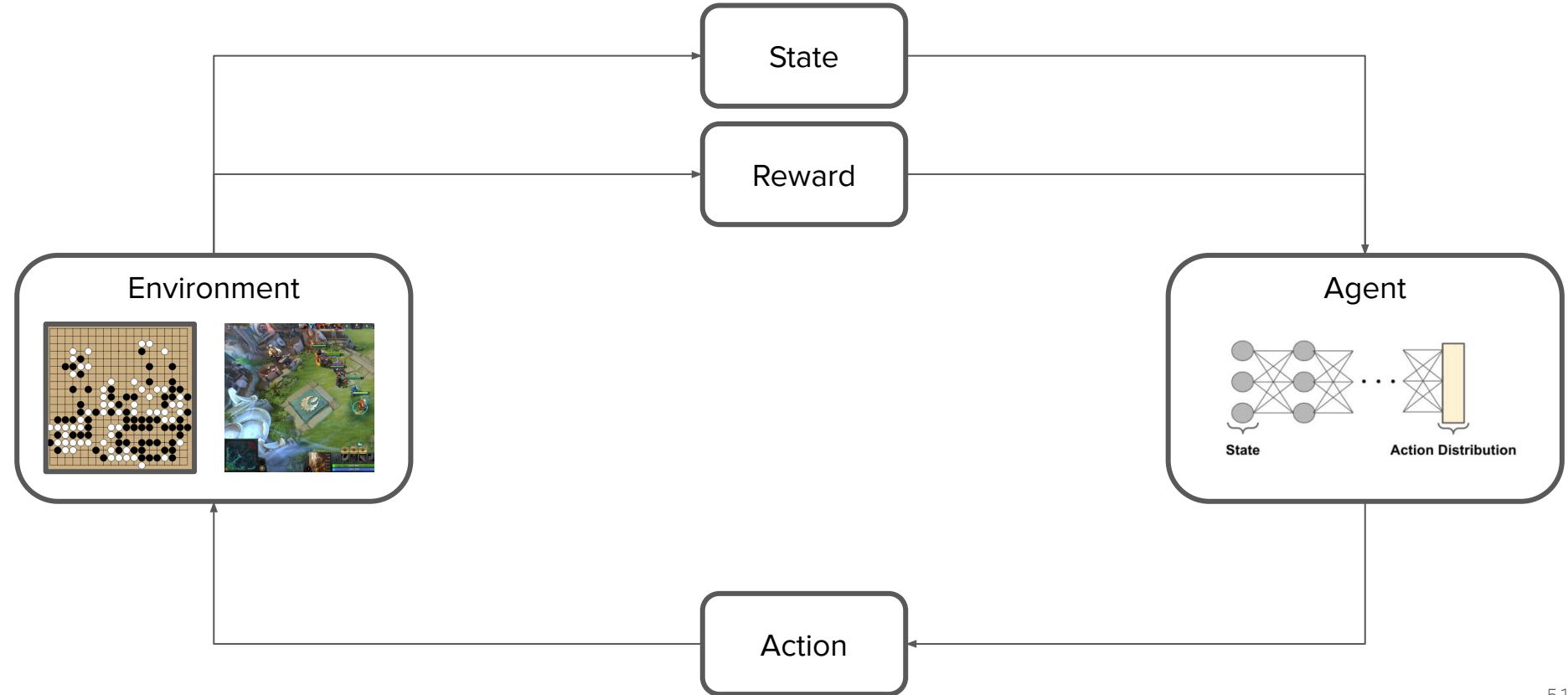
**Shane Legg**  
DeepMind  
[legg@google.com](mailto:legg@google.com)

**Dario Amodei**  
OpenAI  
[damodei@openai.com](mailto:damodei@openai.com)

Source:

[https://proceedings.neurips.cc/paper\\_files/paper/2017/file/d5e2c0adad503c91f91df240d0cd4e49-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2017/file/d5e2c0adad503c91f91df240d0cd4e49-Paper.pdf)

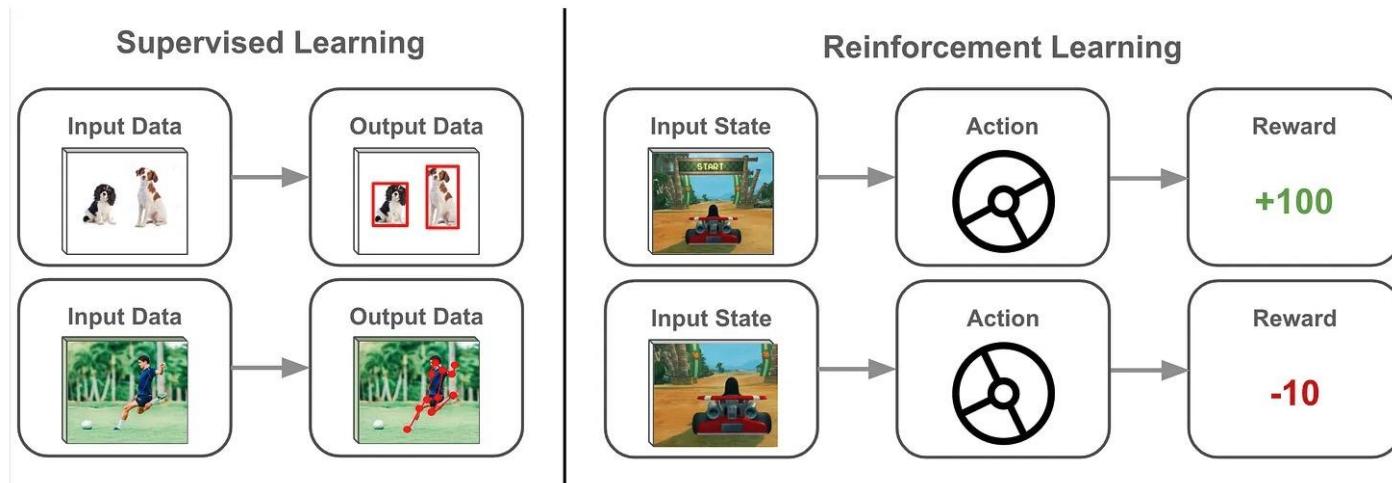
# Framework for RL



# Framework for RL

- Agent & Environment
  - An agent interacts with an environment
  - The agent has a state in the environment
- Actions & State Changes
  - The agent produces actions that modify its state
  - Actions influence future states
- Rewards & Optimization
  - The agent receives positive or negative rewards
  - The goal is to maximize rewards over time

# Framework for RL



# RLHF: Reward Model

- Generate training datasets of prompt&response pairs
  - Sample few prompts from a pre-defined dataset
  - Pass the prompt to the initial language model
  - Collect different responses from the LM (by setting temperature or using different checkpoints)
- Humans annotators are used to rank the generated responses from the LM
  - Higher rank for that pair, higher reward of the response to the prompt

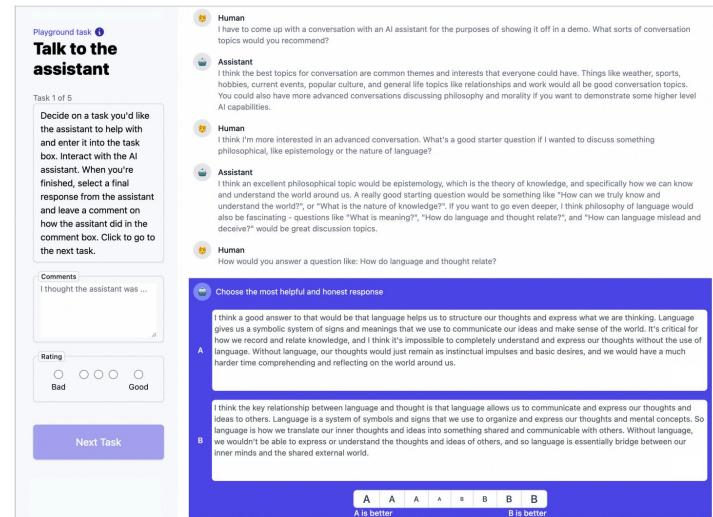
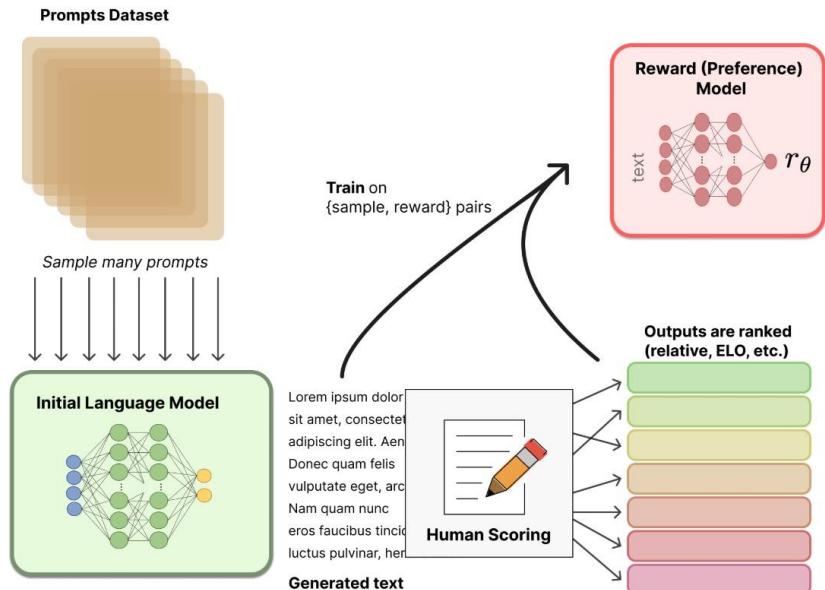


Figure 6 We show the interface that crowdworkers use to interact with our models. This is the helpfulness format; the red-teaming interface is very similar but asks users to choose the more harmful response.

# RLHF: Reward Model

- Reward model is trained from the above pairs.
  - (Prompt, Response) -> Reward Signal
  - Try to learn the preference from humans



<https://huggingface.co/blog/rlhf>

# Reward model (InstructGPT)

- Prepare RM dataset (33k prompts)
  - Given prompts and multiple model outputs, labelers are asked to give ranking
- Train a Reward model
  - Take a prompt and a response, and output a scalar reward
  - Loss function for the reward model:

$$\text{loss}(\theta) = -\frac{1}{\binom{K}{2}} E_{(x, y_w, y_l) \sim D} [\log (\sigma(r_\theta(x, y_w) - r_\theta(x, y_l)))]$$

where  $r_\theta(x, y)$  is the scalar output of the reward model for prompt  $x$  and completion  $y$  with parameters  $\theta$ ,  $y_w$  is the preferred completion out of the pair of  $y_w$  and  $y_l$ , and  $D$  is the dataset of human comparisons.

The list C in this code is used to store the values of the binomial coefficient as the function iterates through the values of n and r. It is used to calculate the value of the binomial coefficient for a given value of n and r, and stores the result in the final return value of the function.

(1)

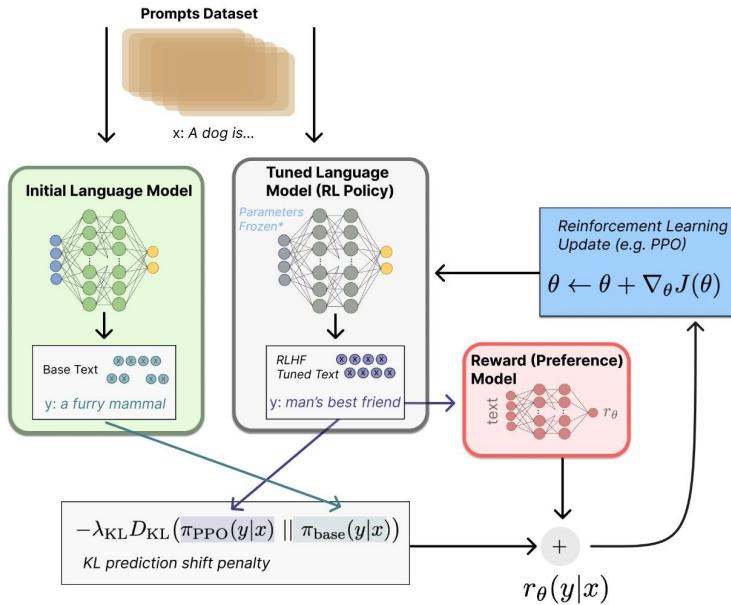
What is the purpose of the list C in the code below?

```
def binomial_coefficient(n, r):
    C = [0 for i in range(r + 1)];
    C[0] = 1;
    for i in range(1, n + 1):
        j = min(i, r);
        while j > 0:
            C[j] += C[j - 1];
            j -= 1;
    return C[r]
```

- A. to store the value of C[0]  
B. to store the value of C[1]  
C. to store the value of C[i]  
D. to store the value of C[i - 1]

# RLHF: Fine-tuning with RL

- Fine-tuning here is formulated as a RL problem
- Parameters of LM are updated to maximize the reward metrics as a combination of the reward output and a constraint on policy shift.
  - Optimize the parameters to make sure that LLM can generate the response which can have a higher reward signal and also at the same time, it is not too far from the original response.
  - More details could be found [here](#)



# RL-tuning (InstructGPT)

- Prepare PPO dataset (31k prompts)
  - PPO: proximal policy optimization (PPO) - a policy-gradient RL algorithm
- Fine-tune SFT model using PPO algorithm to get the preference-tuned model

# How good is RLHF

1.3B RLHF outperform 175B GPT3 on human preferences

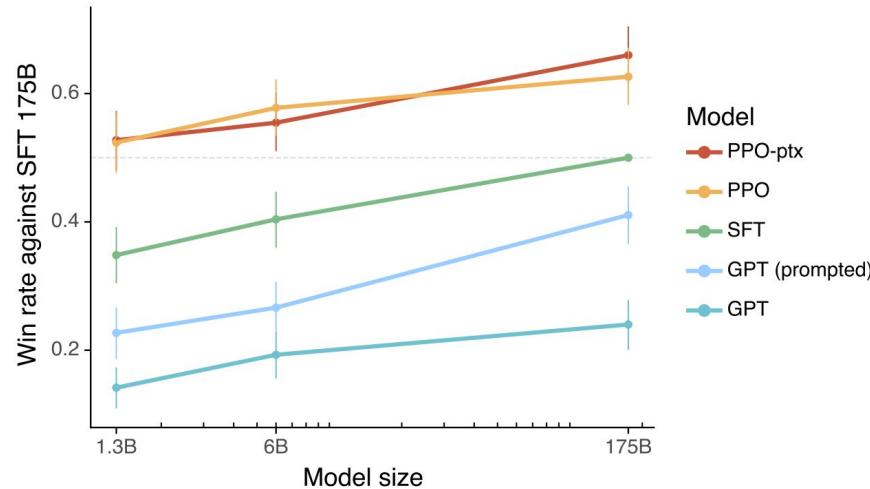


Figure 1: Human evaluations of various models on the API prompt distribution, evaluated by how often outputs from each model were preferred to those from the 175B SFT model. Our InstructGPT models (PPO-ptx) as well as its variant trained without pretraining mix (PPO) significantly outperform the GPT-3 baselines (GPT, GPT prompted); outputs from our 1.3B PPO-ptx model are preferred to those from the 175B GPT-3. Error bars throughout the paper are 95% confidence intervals.

Source

[https://proceedings.neurips.cc/paper\\_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf](https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf)

# Wrap it up

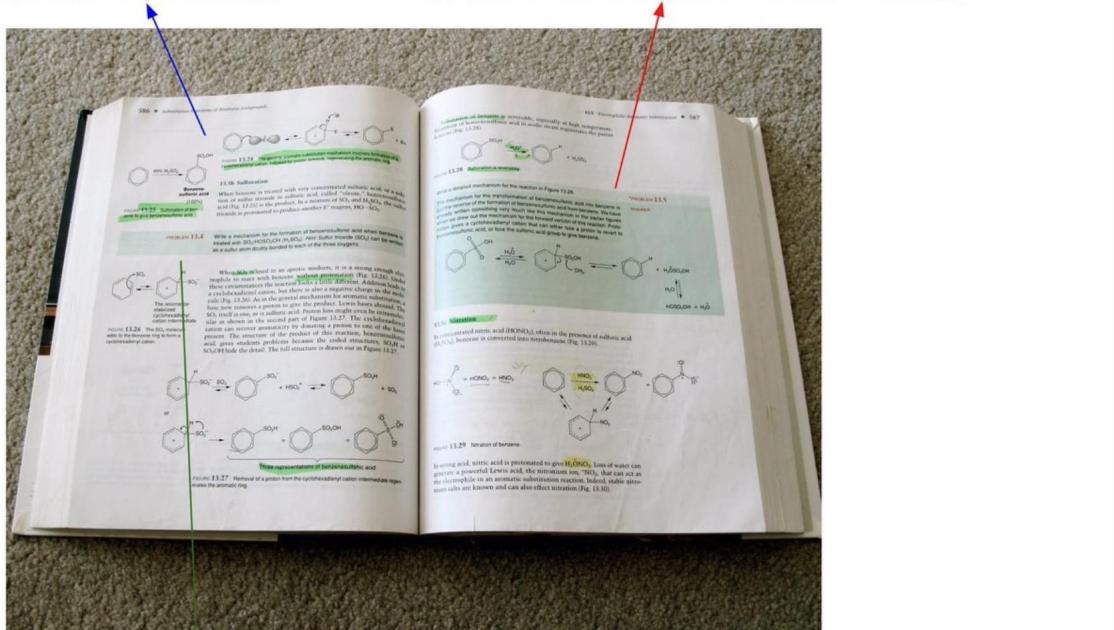
- How to train your LLM from scratch
  - Stage 1: Pretraining
    - Download large scale text data (~10TB)
    - Get a cluster of 6k GPUs
    - Compress the text into the 100 billion parameters and its associated neural network (~\$2M and ~12days)
    - Obtain the foundation/base model
  - Stage 2: Alignment
    - Write labeling instructions
    - Hire ppls, collect 100K high quality prompt responses, and comparisons
    - Finetune the base model on this data
    - Obtain assistant/chatbot model
    - Run a lot of evaluations
    - Deploy, Monitor, Collect misbehaviors



## Post

exposition  $\Leftrightarrow$  pretraining  
(background knowledge)

worked problems  $\Leftrightarrow$  supervised finetuning  
(problem + demonstrated solution, for imitation)



practice problems  $\Leftrightarrow$  reinforcement learning  
(prompts to practice, trial & error until you reach the correct answer)

Source:

[https://x.com/karpathy/status/1885026028428681698?ref\\_src=twsrct%5Egoogle%7Ctgcamp%5Esrc%7Ctwgr%5Etweet](https://x.com/karpathy/status/1885026028428681698?ref_src=twsrct%5Egoogle%7Ctgcamp%5Esrc%7Ctwgr%5Etweet)

# Toy Example: Train a customer service bot

- Pre-training:
  - The model predicts the next token based on massive unstructured text data
  - For example: in human-written texts, different authors might write:
    - I'd like to speak to a manager. I'd like to speak to the support team. I'd like to speak to your representative.
      - **Input:** I would like to speak to
      - **Target Output:** ["a", "the", "your"]

# Toy Example: Train a customer service bot

- SFT:
  - The model is trained on task-specific labeled data, still using next-token prediction but now in a structured way (e.g., customer service replies, legal answers)
  - For example: we asked the experienced customer service team to provide responses
    - Input: “Customer: I’d like to speak to someone about my complaint.  
Agent:”
    - Target output: “Sure! Let me transfer you to a manager”

# Toy Example: Train a customer service bot

- RLHF:
  - The model generates multiple responses, and humans rank them
  - For example:
    - Input: “Customer: I’d like to speak to someone about my complaint.  
Agent:”
    - Target output (ranked by human reviewers):
      - “Sure! Let me transfer you to a manager” (Rank 1)
      - “We don’t handle complaints” (Rank 3)
      - “Please explain your issue first” (Rank 2)
    - Training adjustment:
      - The model learns to favor response #1 and avoid response #2

# A Joke about Fine-tuning

## Fine-tuning GPT3.5-turbo based on 140k slack messages

September 4, 2023 • 5-minute read

AI • GPT • LLMs

USER write a 500 word blog post on prompt engineering

---

ASSISTANT sure  
I shall work on that in the morning

---

USER It's morning now

---

ASSISTANT I'm writing it right now  
It's 6:30am here

---

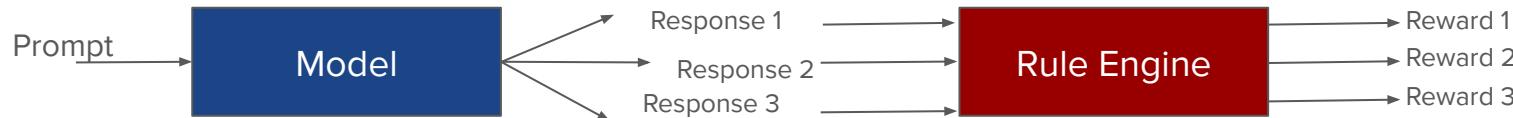
USER Write it now

---

ASSISTANT ok

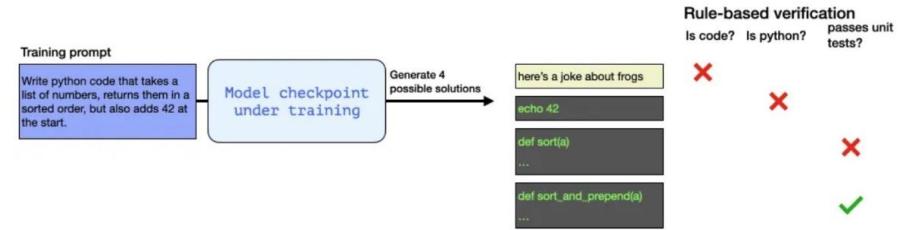
# Rule-based Reinforcement Learning

- It is used by **DeepSeek-R1**
- The reward is automatically calculated by rule-based systems



# Rules to determine reward

- Accuracy:
  - Math: validate answer
  - Coding: auto-validation
- Format: validate output format



Source: <https://newsletter.languageme.models.co/p/the-illustrated-deepseek-r1>

---

A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer. The reasoning process and answer are enclosed within <think> </think> and <answer> </answer> tags, respectively, i.e., <think> reasoning process here </think> <answer> answer here </answer>. User: **prompt**. Assistant:

Table 1 | Template for DeepSeek-R1-Zero. **prompt** will be replaced with the specific reasoning question during training.

Source: <https://arxiv.org/pdf/2501.12948>

Next Class: LLM-RAG

# **Appendix: How to train BERT**

# How to train BERT

Pre-training then connect with a downstream fully-connected layers for fine-tuning

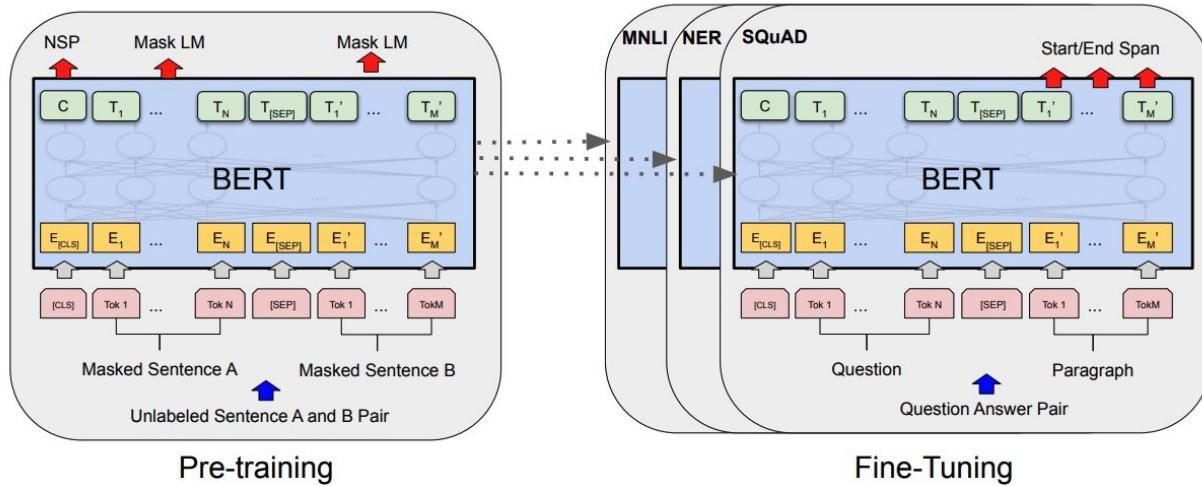


Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token (e.g. separating questions/answers).

source: <https://arxiv.org/pdf/1810.04805.pdf>

# How to Pre-Train



Yann LeCun

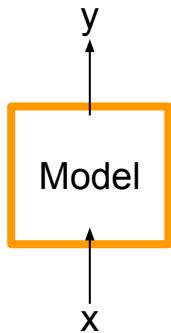
2019年4月30日 · 🌎

...

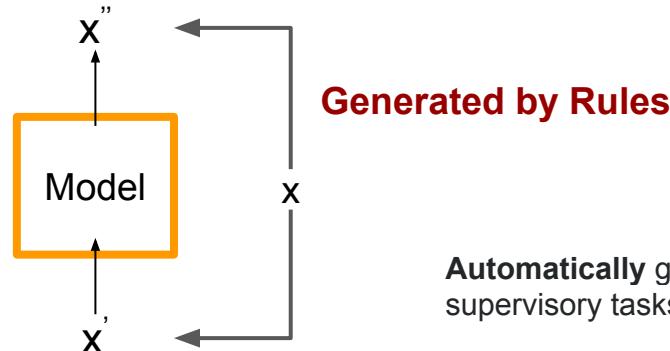
The answer is **self-supervised learning**.

I now call it "self-supervised learning", because "unsupervised" is both a loaded and confusing term.

In self-supervised learning, the system learns to predict part of its input from other parts of it input. In other words a portion of the input is used as a supervisory signal to a predictor fed with the remaining portion of the input.



Supervised



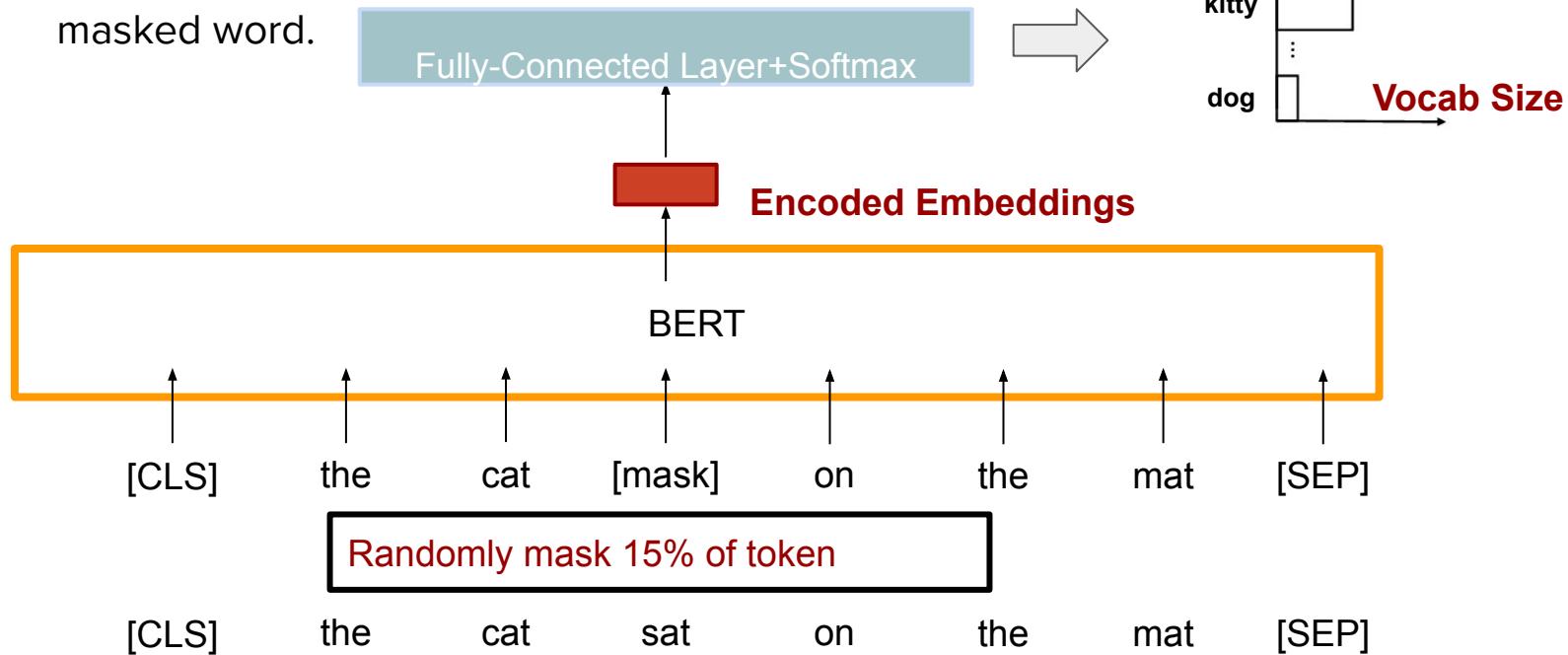
Self-Supervised

Automatically generate some kind of supervisory tasks

# Pre-training Task I: MLM

- **Masked Language Model**

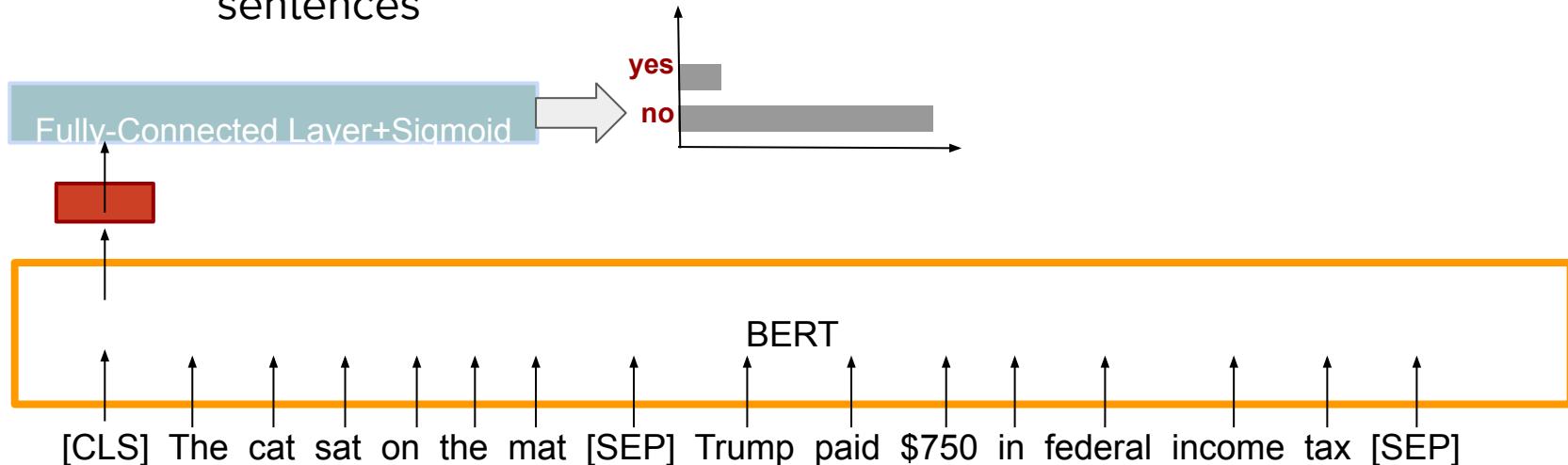
- Use the encoded embeddings of the masked word's to predict the masked word.



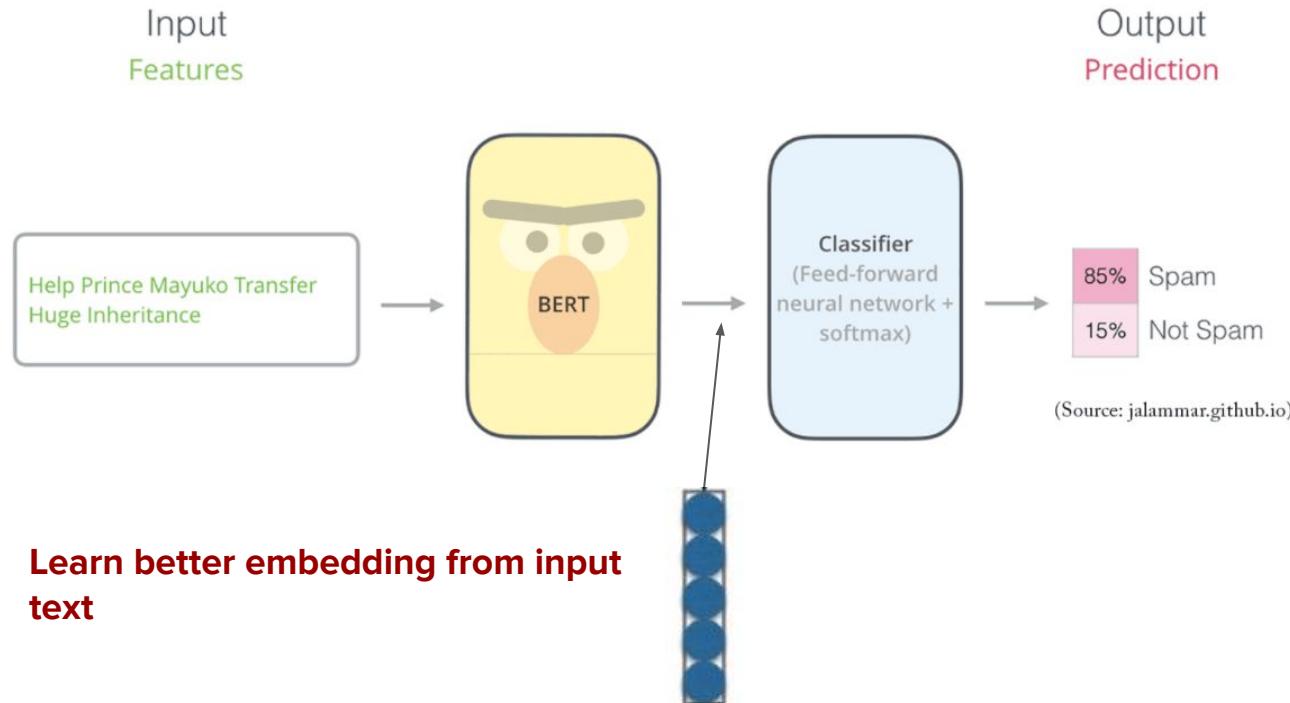
# Pre-training Task II: NSP

- **Next sentence prediction**

- Given two sentences A and B, is B likely to be the sentence followed by A?
- Make bert good at handling relationships between multiple sentences



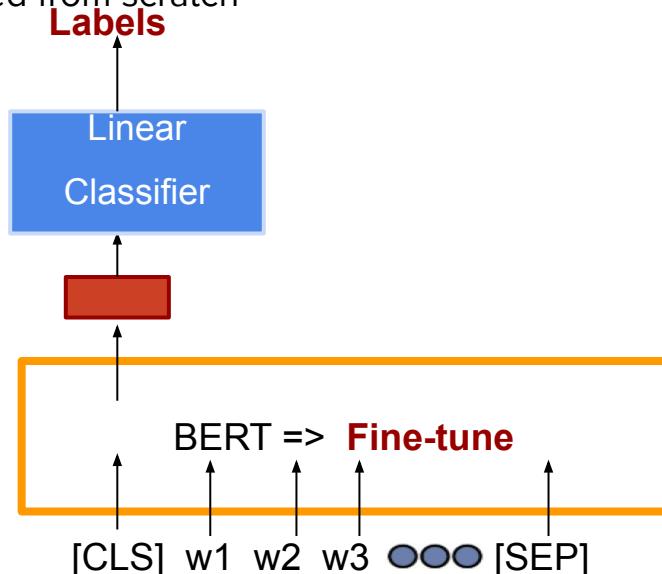
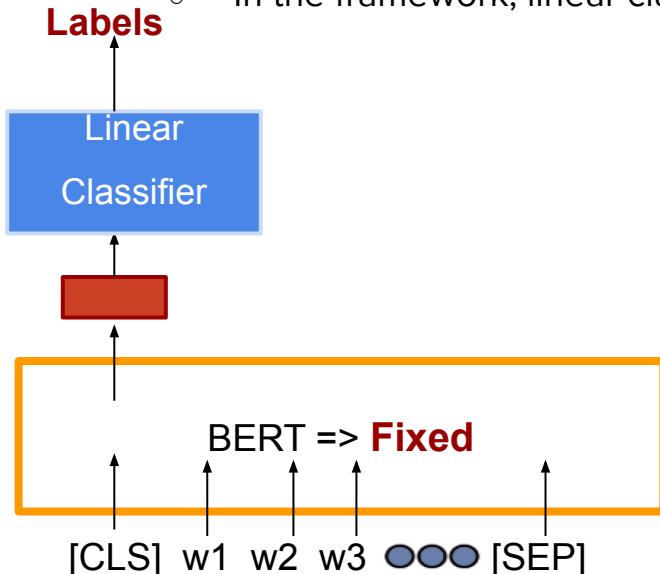
# Bert + Fine-tuning



# How to use BERT - Sequence classification

- **Input: Single Sentence Output: Class**

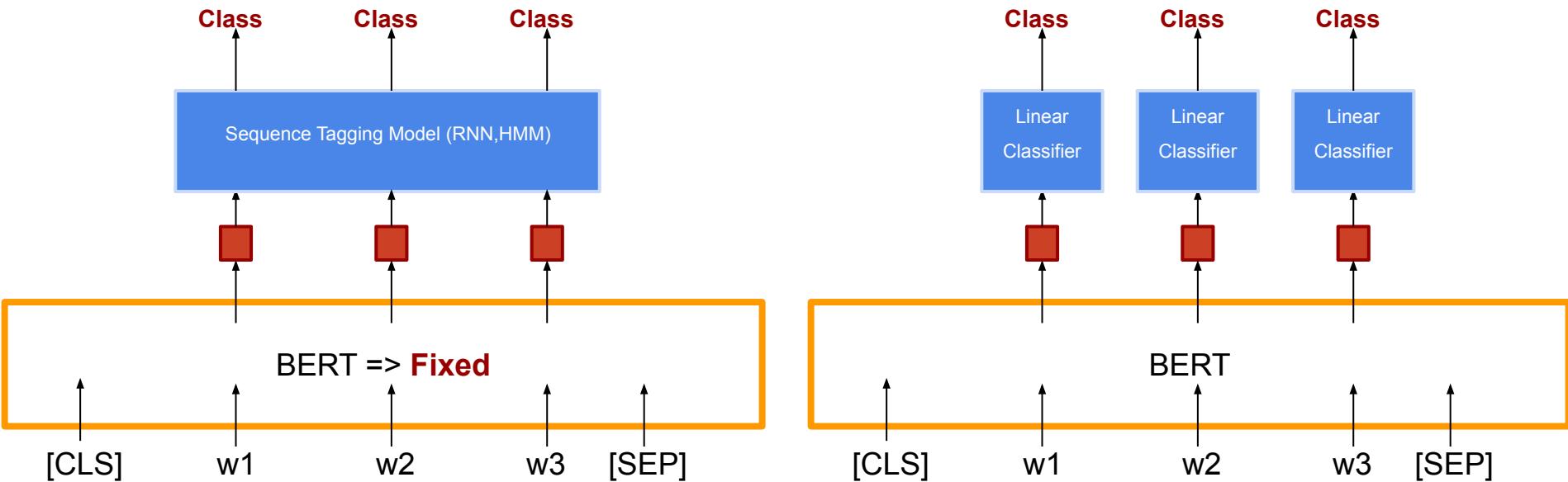
- Sentiment Analysis
- Document Classification
- In the framework, linear classifier should be trained from scratch



# How to use BERT - Sequential Tagging

- **Input: Single Sentence Output: Class**

- NER, POS Tagging



# GPT1

---

## Improving Language Understanding by Generative Pre-Training

---

Alec Radford

OpenAI

alec@openai.com

Karthik Narasimhan

OpenAI

karthikn@openai.com

Tim Salimans

OpenAI

tim@openai.com

Ilya Sutskever

OpenAI

ilyasu@openai.com

### Abstract

Natural language understanding comprises a wide range of diverse tasks such as textual entailment, question answering, semantic similarity assessment, and document classification. Although large unlabeled text corpora are abundant, labeled data for learning these specific tasks is scarce, making it challenging for discriminatively trained models to perform adequately. We demonstrate that large gains on these tasks can be realized by *generative pre-training* of a language model on a diverse corpus of unlabeled text, followed by *discriminative fine-tuning* on each specific task. In contrast to previous approaches, we make use of task-aware input transformations during fine-tuning to achieve effective transfer while requiring minimal changes to the model architecture. We demonstrate the effectiveness of our approach on a wide range of benchmarks for natural language understanding. Our general task-agnostic model outperforms discriminatively trained models that use architectures specifically crafted for each task, significantly improving upon the state of the art in 9 out of the 12 tasks studied. For instance, we achieve absolute improvements of 8.9% on commonsense reasoning (Stories Cloze Test), 5.7% on question answering (RACE), and 1.5% on textual entailment (MultiNLI).

GPT1 laid the groundwork with a decoder-only architecture to show the potential of LLM.

# GPT1 Training

1. Unsupervised Pre-training: next token prediction
2. Fine Tuning: A fully connected layer would be used for label prediction. And it is task-specific

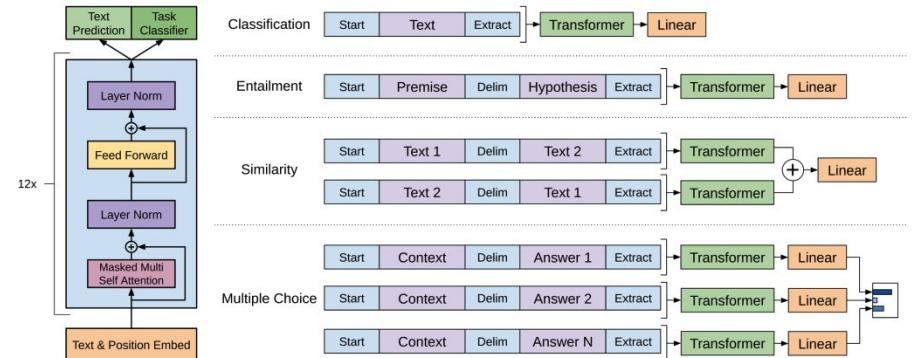


Figure 1: (left) Transformer architecture and training objectives used in this work. (right) Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

Source:

[https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language\\_understanding\\_paper.pdf](https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf)

# GPT2

## Language Models are Unsupervised Multitask Learners

Alec Radford <sup>\*1</sup> Jeffrey Wu <sup>\*1</sup> Rewon Child <sup>1</sup> David Luan <sup>1</sup> Dario Amodei <sup>\*\*1</sup> Ilya Sutskever <sup>\*\*1</sup>

### Abstract

Natural language processing tasks, such as question answering, machine translation, reading comprehension, and summarization, are typically approached with supervised learning on task-specific datasets. We demonstrate that language models begin to learn these tasks without any explicit supervision when trained on a new dataset of millions of webpages called WebText. When conditioned on a document plus questions, the answers generated by the language model reach 55 F1 on the CoQA dataset - matching or exceeding the performance of 3 out of 4 baseline systems without using the 127,000+ training examples. The capacity of the language model is essential to the success of zero-shot task transfer and increasing it improves performance in a log-linear fashion across tasks. Our largest model, GPT-2, is a 1.5B parameter Transformer that achieves state of the art results on 7 out of 8 tested language modeling datasets in a zero-shot setting but still underfits WebText. Samples from the model reflect these improvements and contain coherent paragraphs of text. These findings suggest a promising path towards building language processing systems which learn to perform tasks from their naturally occurring demonstrations.

competent generalists. We would like to move towards more general systems which can perform many tasks – eventually without the need to manually create and label a training dataset for each one.

The dominant approach to creating ML systems is to collect a dataset of training examples demonstrating correct behavior for a desired task, train a system to imitate these behaviors, and then test its performance on independent and identically distributed (IID) held-out examples. This has served well to make progress on narrow experts. But the often erratic behavior of captioning models (Lake et al., 2017), reading comprehension systems (Jia & Liang, 2017), and image classifiers (Alcorn et al., 2018) on the diversity and variety of possible inputs highlights some of the shortcomings of this approach.

Our suspicion is that the prevalence of single task training on single domain datasets is a major contributor to the lack of generalization observed in current systems. Progress towards robust systems with current architectures is likely to require training and measuring performance on a wide range of domains and tasks. Recently, several benchmarks have been proposed such as GLUE (Wang et al., 2018) and decaNLP (McCann et al., 2018) to begin studying this.

Multitask learning (Caruana, 1997) is a promising framework for improving general performance. However, multitask training in NLP is still nascent. Recent work reports modest performance improvements (Yogatama et al.,

1. GPT2 is trying to build a general language model that could do multi-task learning while training
2. Compared to GPT1, there is no change in architecture. GPT2 has more parameters and a much bigger training dataset
3. No fine-tuning

# GPT2

"I'm not the cleverest man in the world, but like they say in French: **Je ne suis pas un imbecile [I'm not a fool]**.

In a now-deleted post from Aug. 16, Soheil Eid, Tory candidate in the riding of Joliette, wrote in French: "**Mentez mentez, il en restera toujours quelque chose**," which translates as, "Lie lie and something will always remain."

"I hate the word '**perfume**'," Burr says. 'It's somewhat better in French: '**parfum**'.

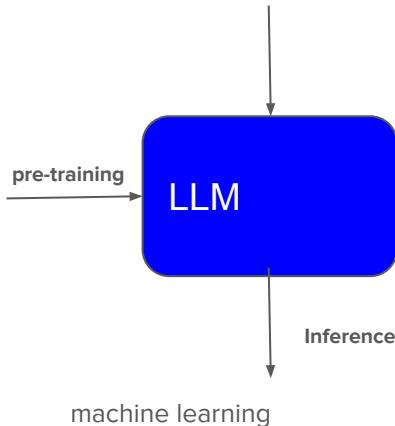
If listened carefully at 29:55, a conversation can be heard between two guys in French: "**-Comment on fait pour aller de l'autre côté? -Quel autre côté?**", which means "**- How do you get to the other side? - What side?**".

If this sounds like a bit of a stretch, consider this question in French: **As-tu aller au cinéma?**, or **Did you go to the movies?**, which literally translates as Have-you to go to movies/theater?

**"Brevet Sans Garantie Du Gouvernement"**, translated to English: "**Patented without government warranty**".

*Table 1. Examples of naturally occurring demonstrations of English to French and French to English translation found throughout the WebText training set.*

深度学习=deep learning  
商业分析=business analytics  
机器学习=



A context of example pairs of chinese text=english is provided to help the LLM infer this is the machine translation task.

# Downstream NLP tasks can all be formulated as LM

- Language model is doing next token prediction
  - E.g., based on the previous tokens: I love this -> movie
- Downstream NLP tasks:
  - Sentiment analysis: given a sentence, **generate** sentiment label
    - I love this movie -> positive
  - Machine translation: given a source sentence, **generate** a target sentence
    - 深度学习 -> deep learning
- How LM differentiate those NLP tasks?
  - Provide in-context information (prompt)

深度学习 Translate it into english:

I love this movie. Label it sentiment

