

# **Applied Machine Learning for Business Analytics**

Lecture 4: LLM I

# Agenda

1. Intro to LLM
2. GPT
3. Scaling Law
4. From GPT to ChatGPT

# 1. Intro to LLM

TABLE I: High-level Overview of Popular Language Models

Type	Model Name	#Parameters	Release	Base Models	Open Source	#Tokens	Training dataset
Encoder-Only	BERT	110M, 340M	2018	-	✓	137B	BooksCorpus, English Wikipedia
	RoBERTa	355M	2019	-	✓	2.2T	BooksCorpus, English Wikipedia, CC-NEWS, STORIES (a subset of Common Crawl), Reddit BooksCorpus, English Wikipedia
	ALBERT	12M, 18M, 60M, 235M	2019	-	✓	137B	BooksCorpus, English Wikipedia, Reddit content
	DeBERTa	-	2020	-	✓	-	BooksCorpus, English Wikipedia, Giga5, Common Crawl, ClueWeb 2012-B
Decoder-only	GPT-1	120M	2018	-	✓	1.3B	BooksCorpus
	GPT-2	1.5B	2019	-	✓	10B	Reddit outbound
Encoder-Decoder	T5 (Base)	223M	2019	-	✓	156B	Common Crawl
	MT5 (Base)	300M	2020	-	✓	-	New Common Crawl-based dataset in 101 languages (m Common Crawl)
	BART (Base)	139M	2019	-	✓	-	Corrupting text
GPT Family	GPT-3	125M, 350M, 760M, 1.3B, 2.7B, 6.7B, 13B, 175B	2020	×	300B	Common Crawl (filtered), WebText2, Books1, Books2, Wikipedia	
	CODEX	12B	2021	GPT	✓	-	Public GitHub software repositories
	WebGPT	760M, 13B, 175B	2021	GPT-3	×	-	ELIS
	GPT-4	1.76T	2023	-	×	13T	-
LLaMA Family	LLaMA1	7B, 13B, 33B, 65B	2023	-	✓	1T, 1.4T	Online sources
	LLaMA2	7B, 13B, 34B, 70B	2023	-	✓	2T	Online sources
	Alpaca	7B	2023	LLaMA1	✓	-	GPT-3.5
	Vicuna-13B	13B	2023	LLaMA1	✓	-	GPT-3.5
	Koala	13B	2023	LLaMA	✓	-	Dialogue data
	Mistral-7B	7.3B	2023	-	✓	-	-
	Code Llama	34	2023	LLaMA2	✓	500B	Publicly available code
	LongLLaMA	3B, 7B	2023	OpenLLaMA	✓	1T	-
	LLaMA-Pro-8B	8.3B	2024	LLaMA2-7B	✓	80B	Code and math corpora
	TinyLlama-1.1B	1.1B	2024	LLaMA1.1B	✓	3T	SlimPajama, Starcoderdata
PaLM Family	PaLM	8B, 62B, 540B	2022	-	×	780B	Web documents, books, Wikipedia, conversations, GitHub code
	U-PaLM	8B, 62B, 540B	2022	-	×	1.3B	Web documents, books, Wikipedia, conversations, GitHub code
	PaLM-2	340B	2023	-	✓	3.6T	Web documents, books, code, mathematics, conversational data
	Med-PaLM	540B	2022	PaLM	×	780B	HealthSearchQA, MedicationQA, LiveQA
	Med-PaLM 2	-	2023	PaLM 2	×	-	MedQA, MedMCQA, HealthSearchQA, LiveQA, MedicationQA
Other Popular LLMs	FLAN	137B	2021	LaMDA-PT	✓	-	Web documents, code, dialog data, Wikipedia
	Gopher	280B	2021	-	×	300B	MassiveText
	ERNIE 4.0	10B	2023	-	×	4TB	Chinese text
	Retro	7.5B	2021	-	×	600B	MassiveText
	LaMDA	137B	2022	-	×	168B	public dialog data and web documents
	ChinChilla	70B	2022	-	×	1.4T	MassiveText
	Galactia-120B	120B	2022	-	-	450B	-
	CodeGen	16.1B	2022	-	✓	-	THE PILE, BIGQUERY, BIGPYTHON
	BLOOM	176B	2022	-	✓	366B	ROOTS
	Zephyr	7.24B	2023	Mistral-7B	✓	800B	Synthetic data
	Grok-0	33B	2023	-	×	-	Online source
	ORCA-2	13B	2023	LLaMA2	-	2001B	GitHub
	StartCoder	15.5B	2023	-	✓	35B	RedPajama, m Common Crawl, S2ORC, Common Crawl
	MPT	7B	2023	-	✓	1T	-
	Mixtral-8x7B	46.7B	2023	-	✓	-	Instruction dataset
	Falcon 180B	180B	2023	-	✓	3.5T	RefinedWeb
	Gemini	1.8B, 3.25B	2023	-	✓	-	Web documents, books, and code, image data, audio data, video data
	DeepSeek-Coder	1.3B, 6.7B, 33B	2024	-	✓	2T	GitHub's Markdown and StackExchange
	DocLLM	1B, 7B	2024	-	×	2T	IIT-CDIP Test Collection 1.0, DocBank

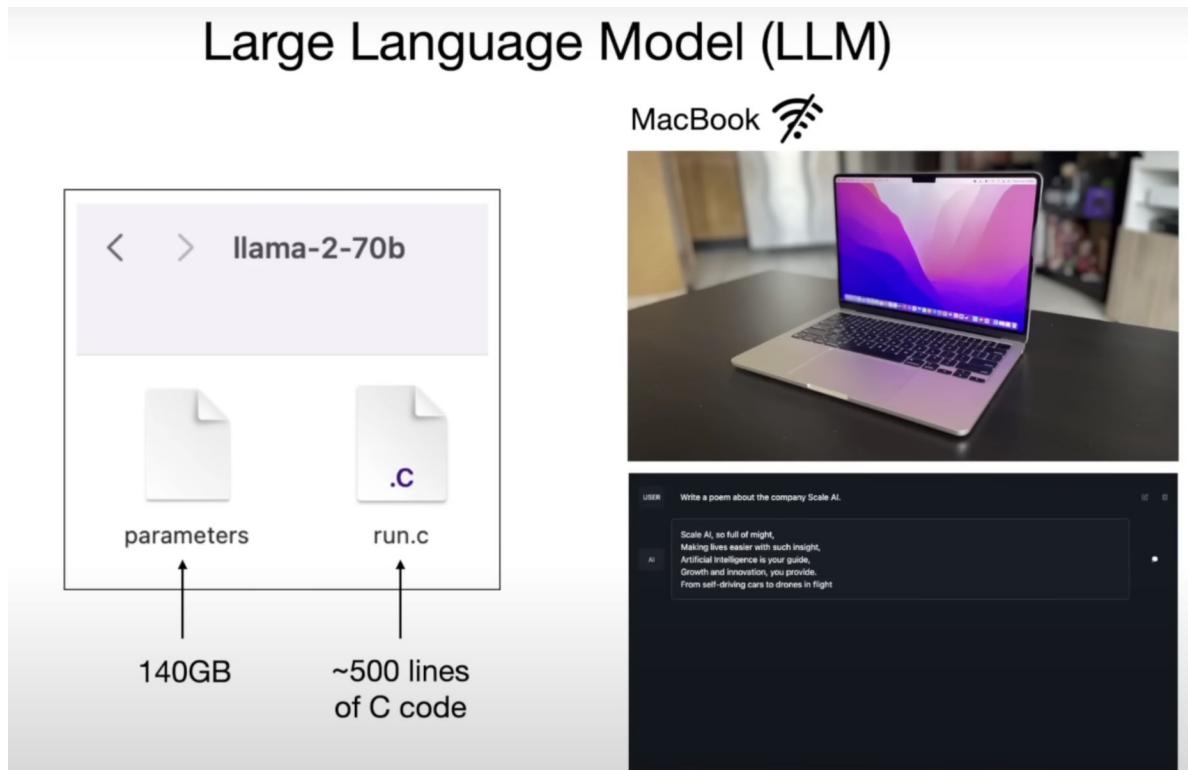
# What is LLM

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

# 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



parameters.zip

Internet data  
~45TB of text

1024 A100 GPUs, 34 days  
\$5Mdata

GPT3 ~175B

# LLM's Evolution Process

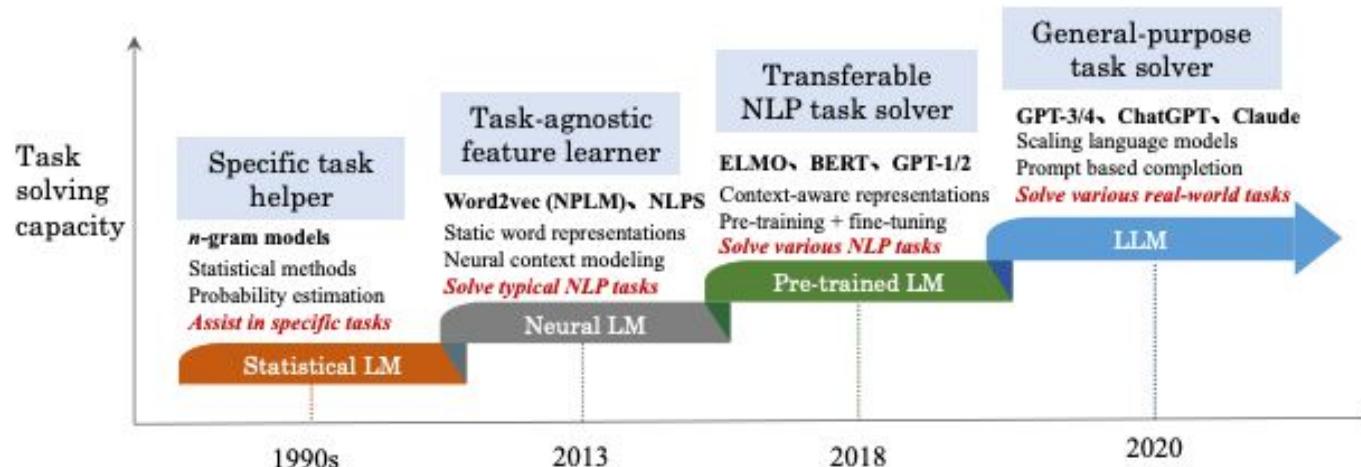
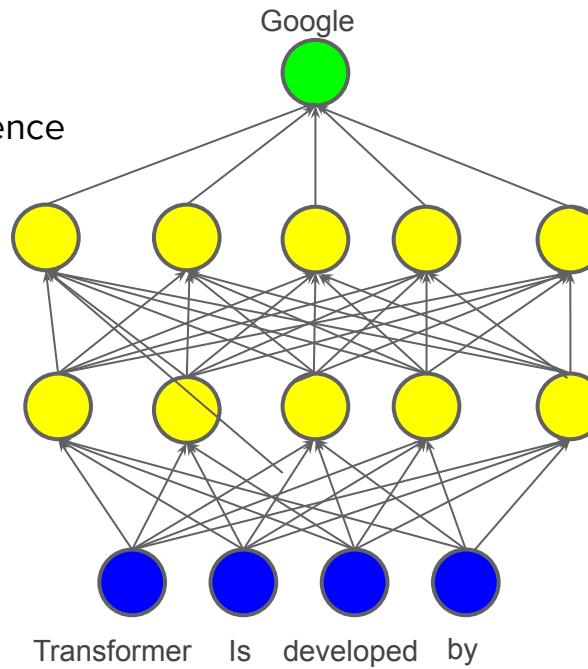


Fig. 2: An evolution process of the four generations of language models (LM) from the perspective of task solving capacity. Note that the time period for each stage may not be very accurate, and we set the time mainly according to the publish date of the most representative studies at each stage. For 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



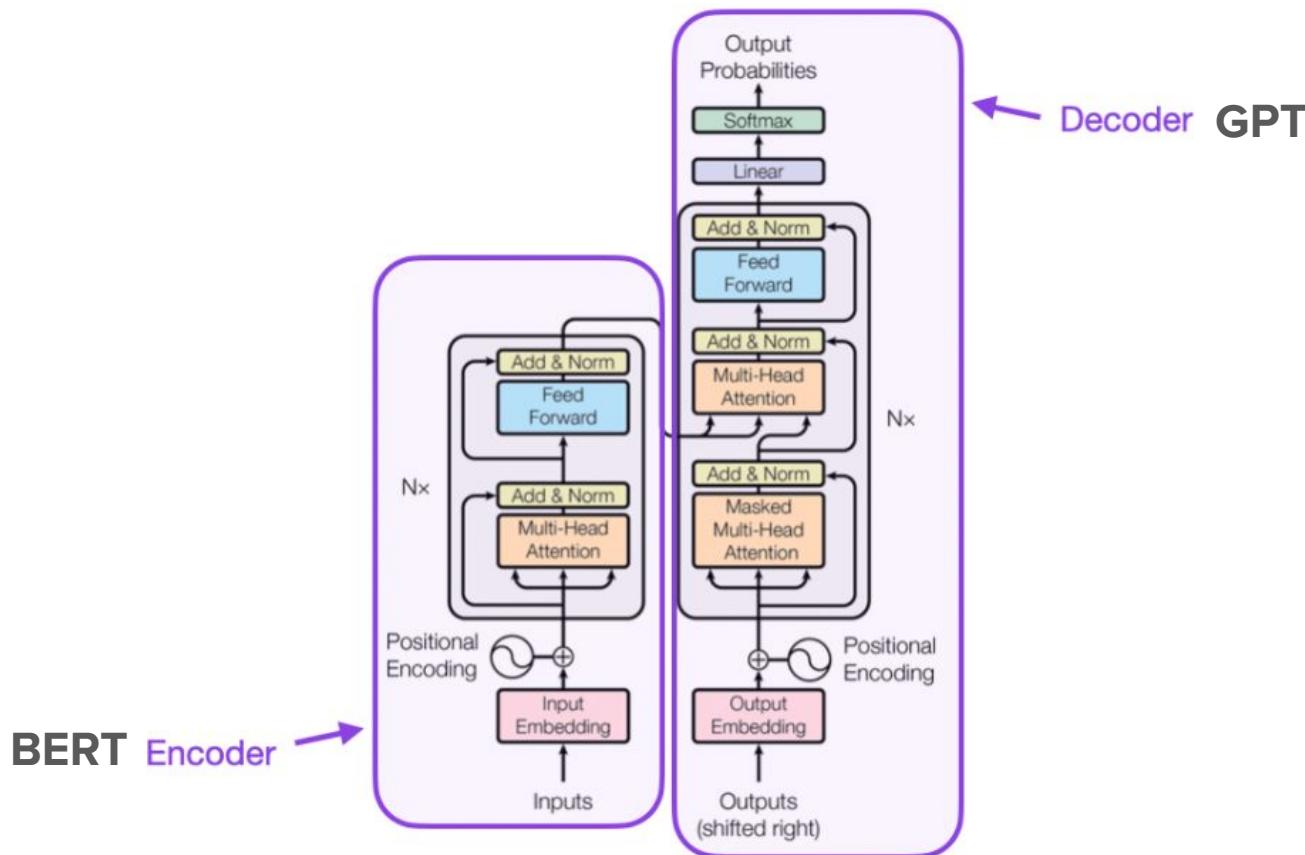
# Word prediction is absorbing knowledge

*"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>

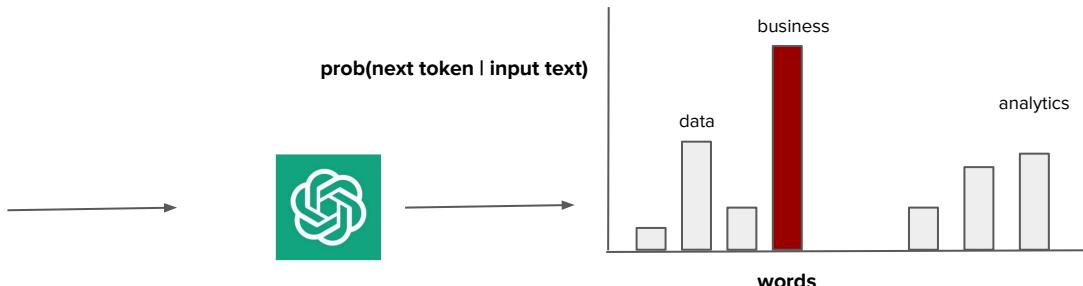
This architecture is now used not only in [natural language processing](#) and [computer vision](#),<sup>[9]</sup> but also in [audio](#)<sup>[10]</sup> and multi-modal processing. It has also led to the development of [pre-trained systems](#), such as [generative pre-trained transformers](#) (GPTs)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

# Transformer is the backbone of LLMs



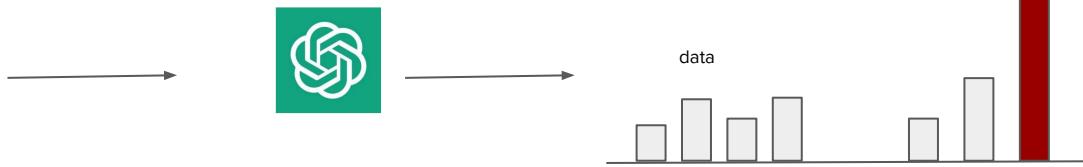
# How does it work

What is business analytics?



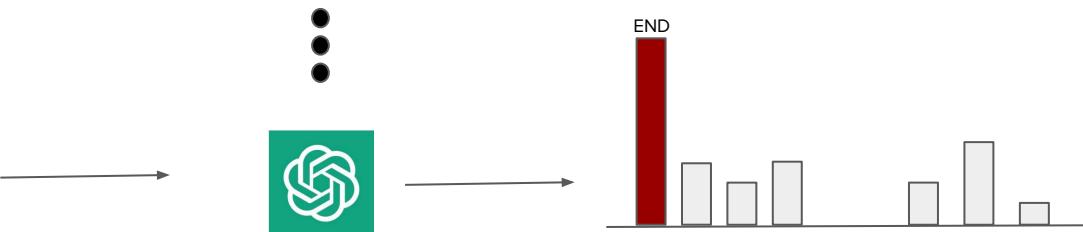
Business

What is business analytics? Business



analytics

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



END

# How does it work

- After pre-training, LLM is kind of knowledge database, but it is a total 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'?

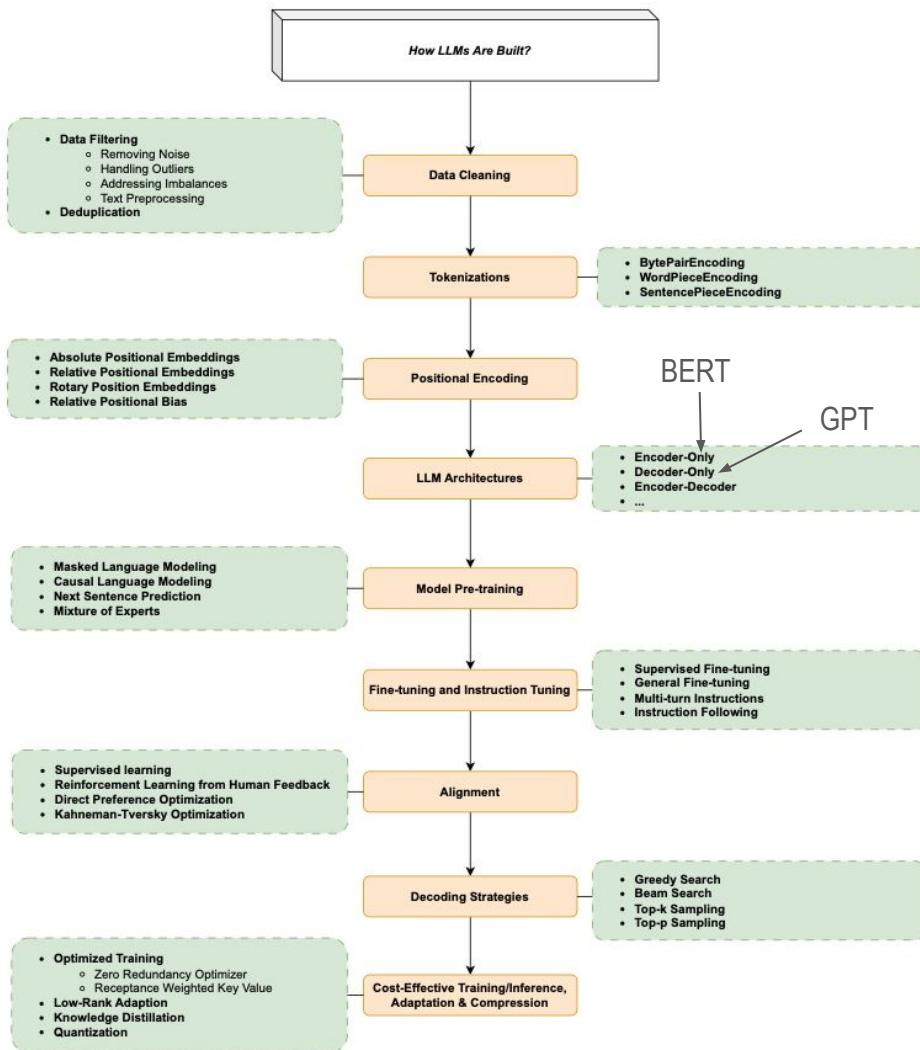


There is one 'm' in the word 'Weather'.



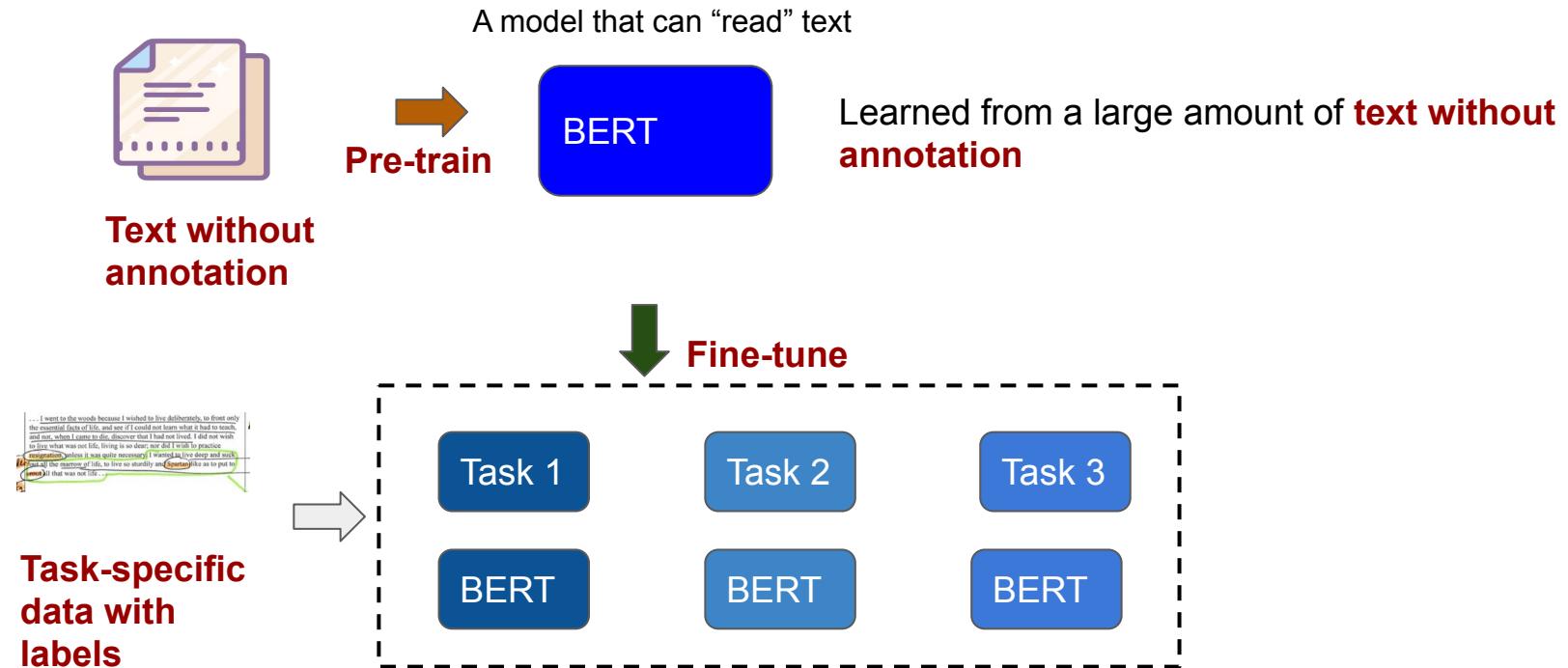
Figure 1: **Inconsistent knowledge in GPT-4.** GPT-4 correctly gives the name of Tom Cruise's mother (left). Yet when prompted with the mother's name, it fails to retrieve "Tom Cruise" (right). We hypothesize this ordering effect is due to the Reversal Curse. Models trained on "A is B" (e.g. "Tom Cruise's 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>

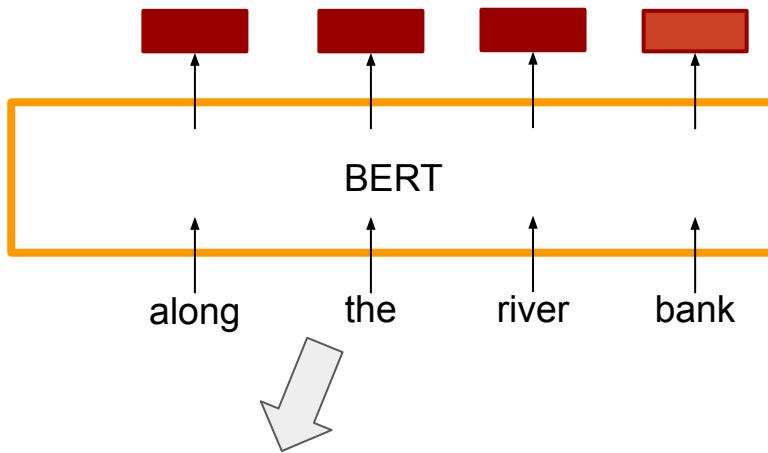
# How to train BERT



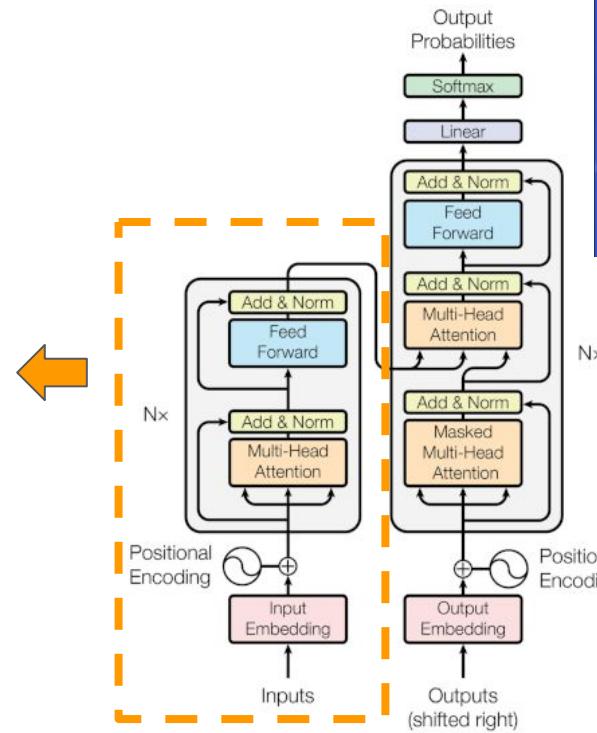
## **2. GPT**

# 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

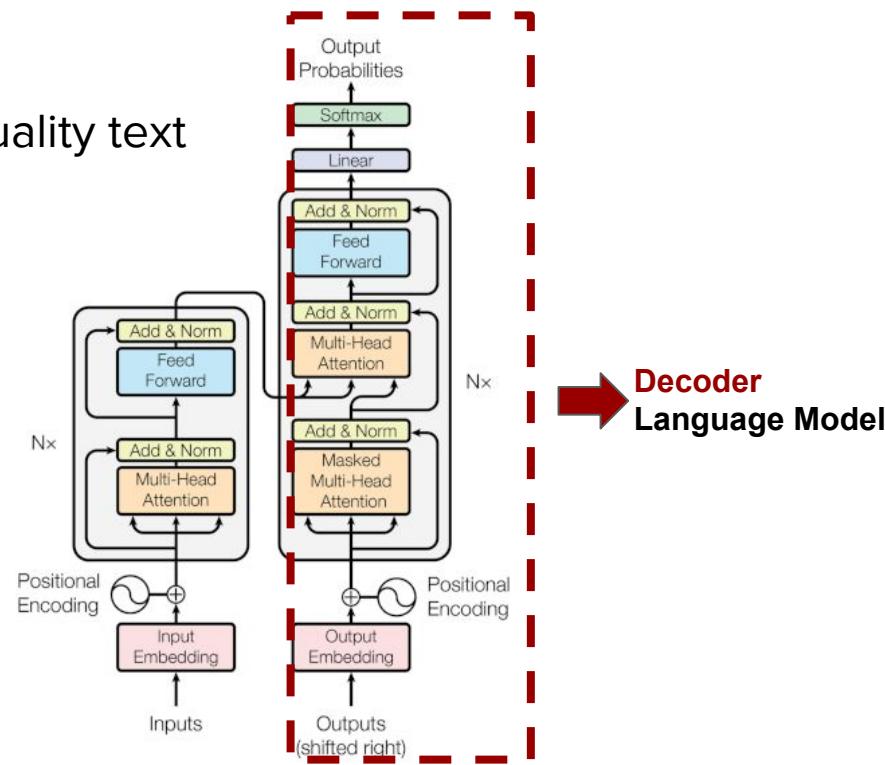


Transformer

Solve Seq2Seq Task <sup>17</sup>

# What is GPT

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



# 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

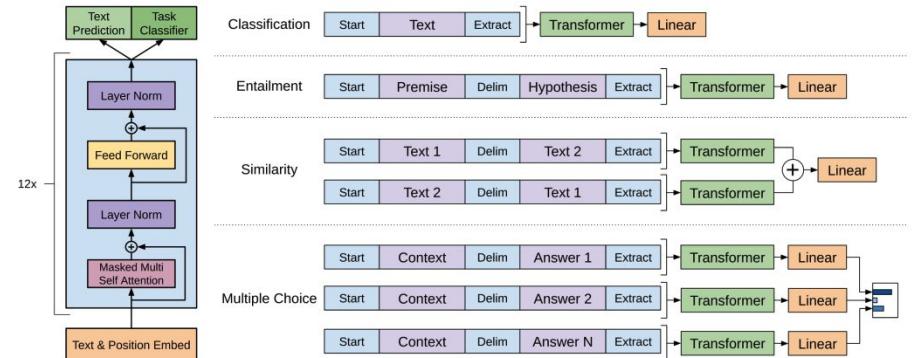


Figure 1: (left) Transformer architecture and training objectives used in this work. (right) Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

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**'.

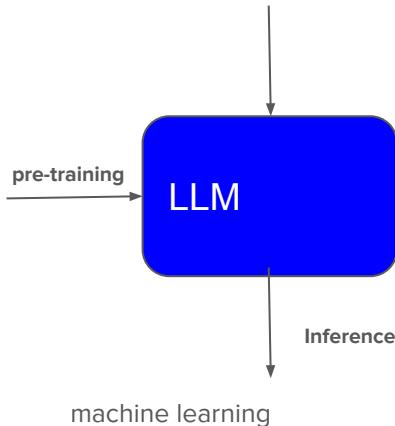
If listened carefully at 29:55, a conversation can be heard between two guys in French: "-**Comment on fait pour aller de l'autre côté? -Quel autre côté?**", which means "- How do you get to the other side? - What side?".

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=



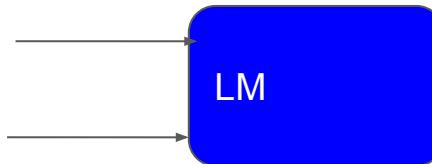
A context of example pairs of chinese text=english is provided to help the LLM infer this is the machine translation task.

# 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



# 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

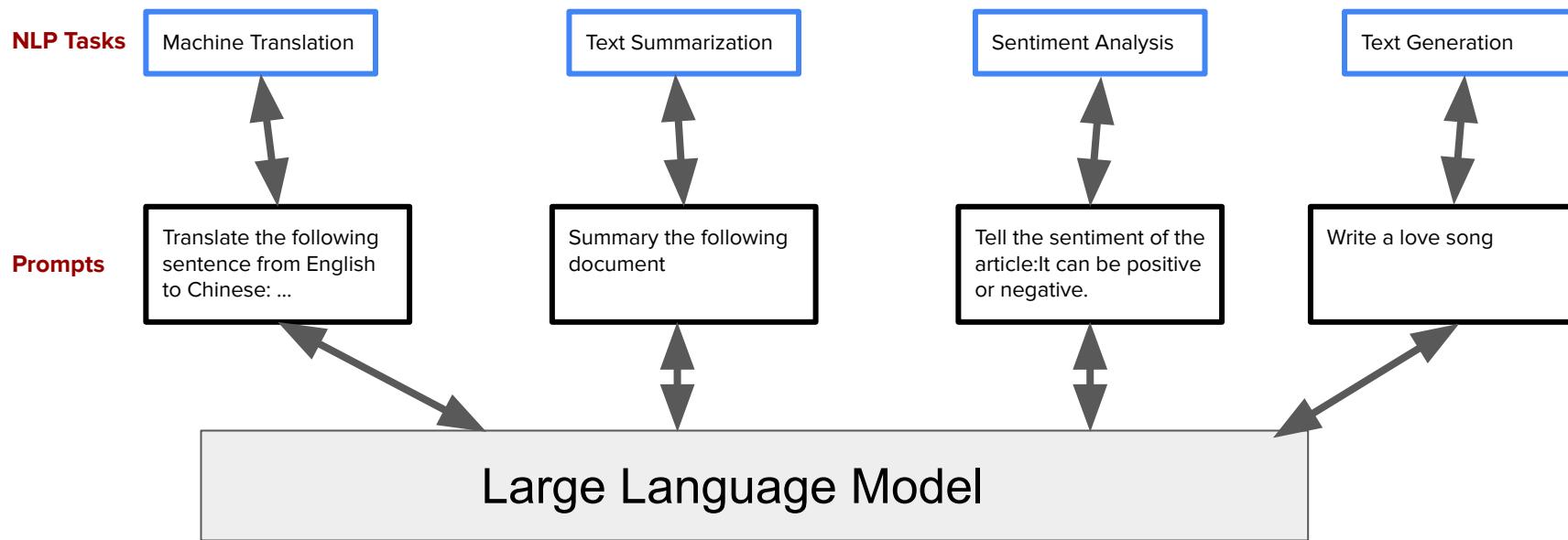
### Abstract

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, 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

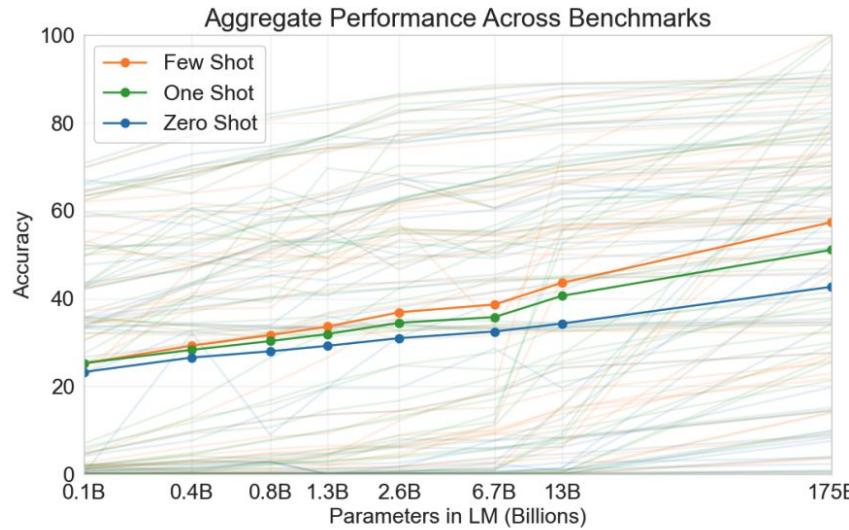
In-context learning: using the text input of a pre-trained language model as a form of task specification: the model is conditioned on a natural language instruction and/or a few demonstrations of the task and is then expected to complete further instances of the task simply by predicting what comes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

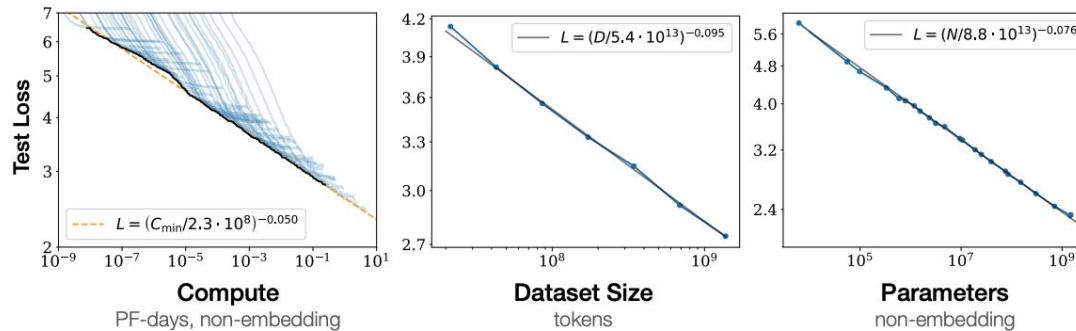


About 587,000,000 results (0.37 seconds)

## 4. 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

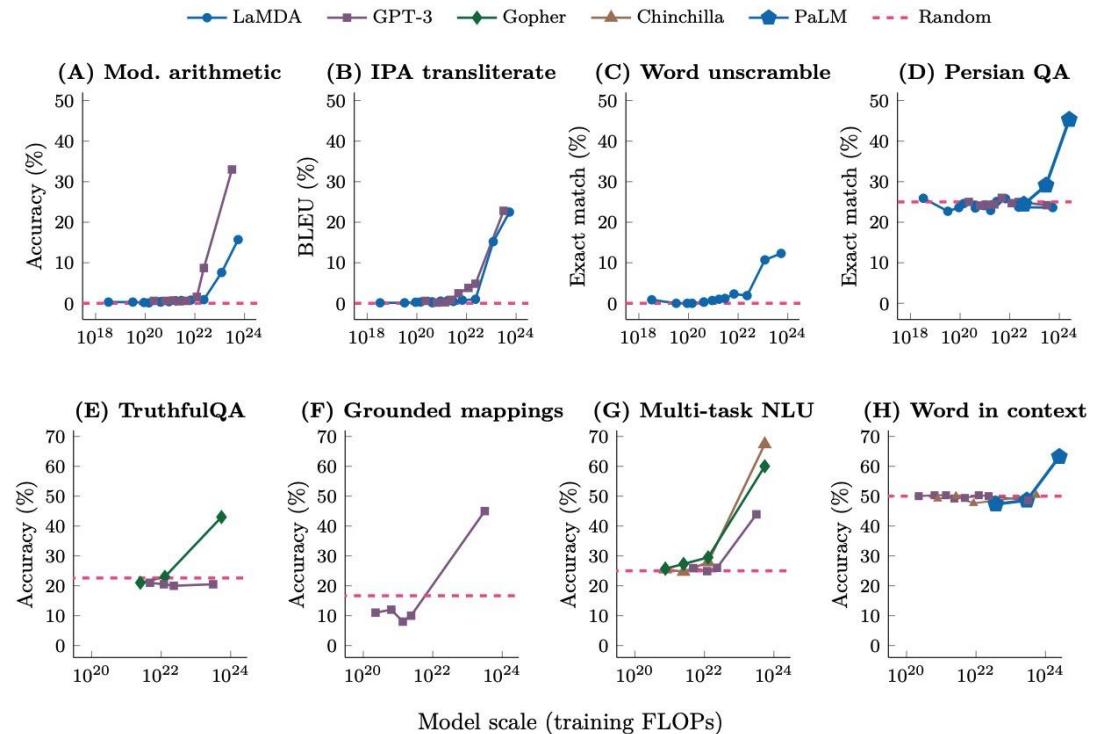


**Figure 1** Language modeling performance improves smoothly as we increase the model size, dataset size, and amount of compute<sup>2</sup> used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual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

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

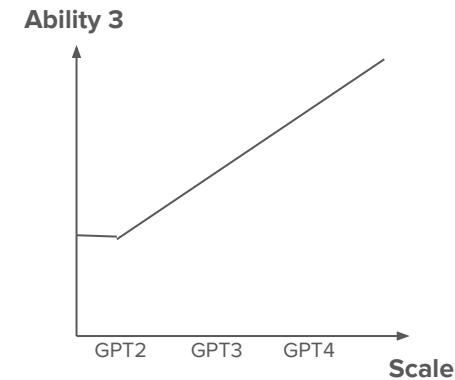
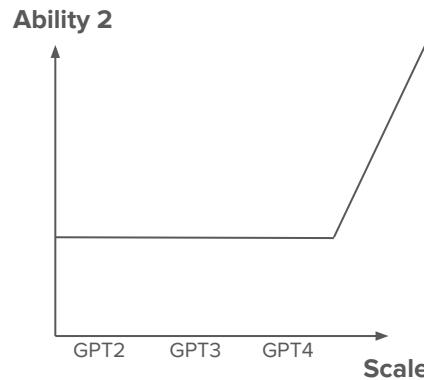
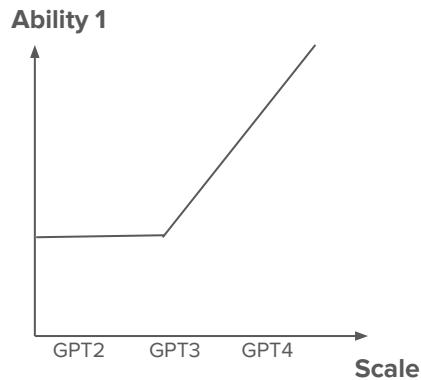
# Emergent abilities of large language models

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



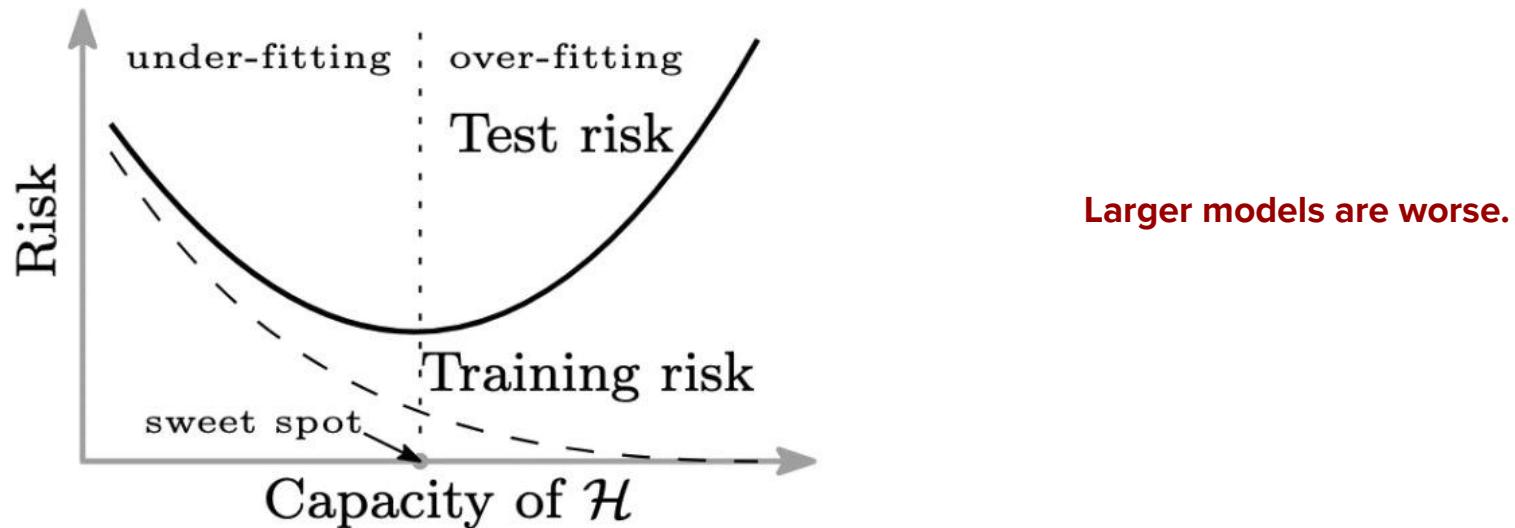
Source: <https://arxiv.org/abs/2206.07682>

# 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.

# Conventional Machine Learning



A model with zero training error is overfitting to the training data and will usually generalize poorly.

# Double Descent

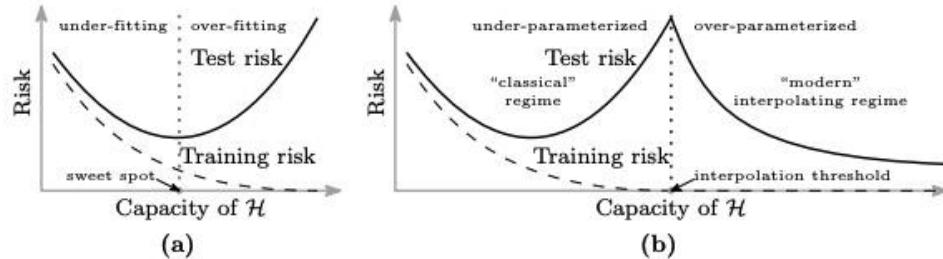


Figure 1: Curves for training risk (dashed line) and test risk (solid line). (a) The classical *U-shaped risk curve* arising from the bias-variance trade-off. (b) The *double descent risk curve*, which incorporates the U-shaped risk curve (i.e., the “classical” regime) together with the observed behavior from using high capacity function classes (i.e., the “modern” interpolating regime), separated by the interpolation threshold. The predictors to the right of the interpolation threshold have zero training risk.

Source: <https://arxiv.org/pdf/1812.11118.pdf>

When the model size is beyond a certain threshold i.e. interpolation threshold, the test error would be in downward trend

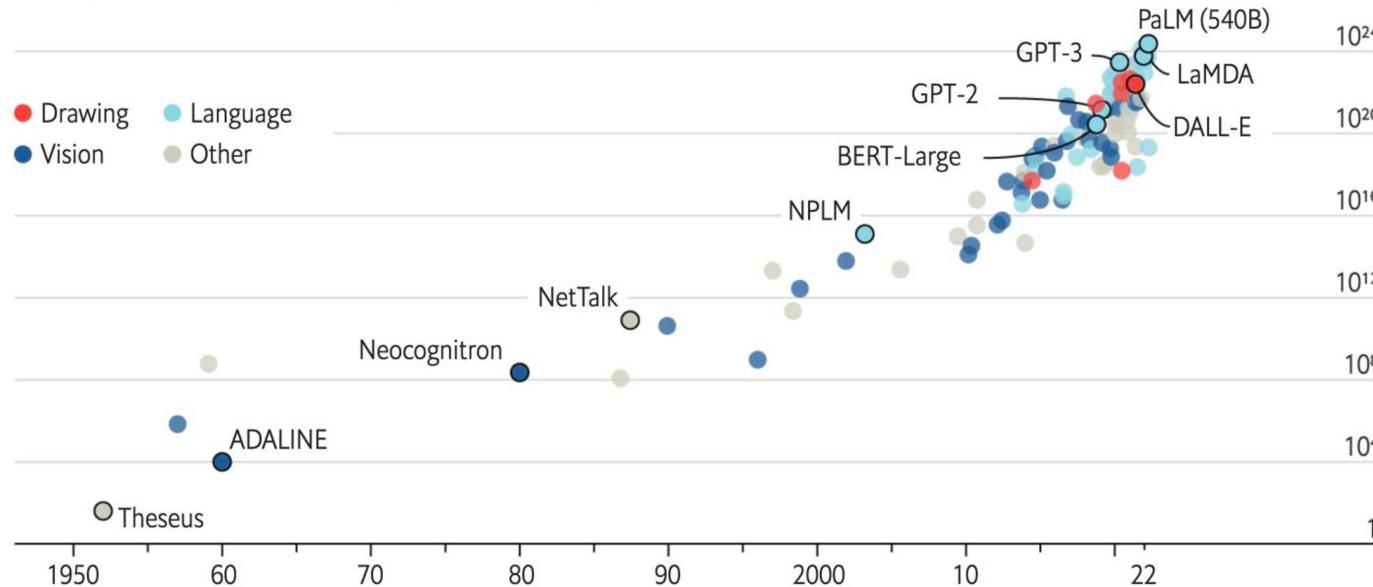
**The threshold  $\approx$  The number of class times the number of training examples**

# Pre-training

## The blessings of scale

AI training runs, estimated computing resources used

Floating-point operations, selected systems, by type, log scale



Sources: "Compute trends across three eras of machine learning", by J. Sevilla et al., arXiv, 2022; Our World in Data

# 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



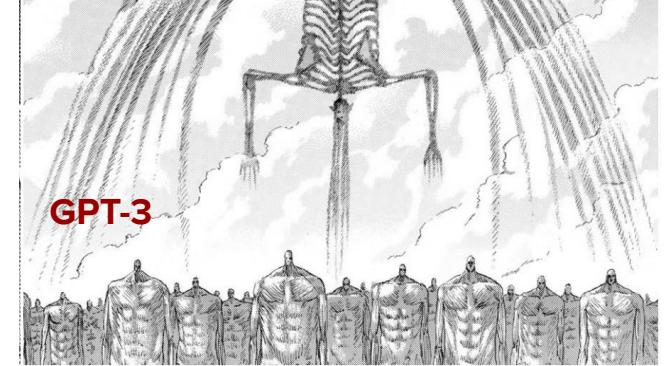
Cart Titan 車力の巨人 4m

BERT-Large

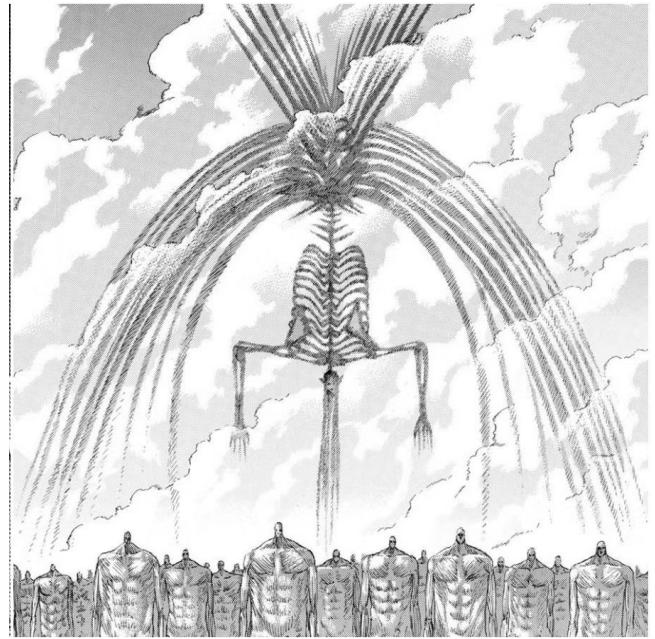


Beast Titan 獣の巨人 17m

GPT-2



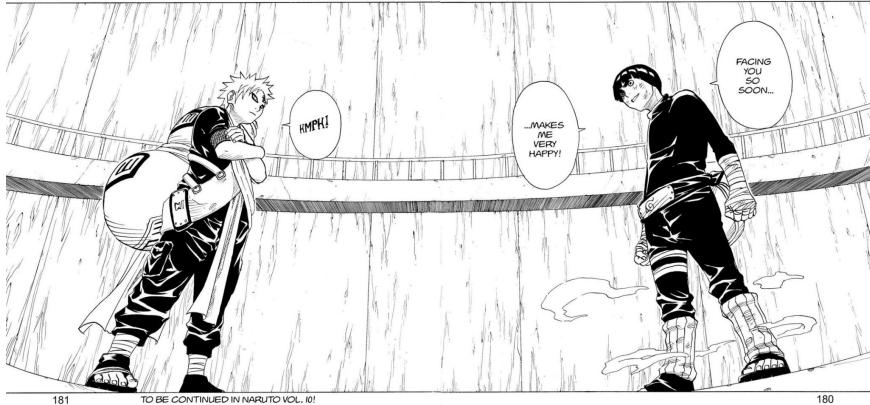
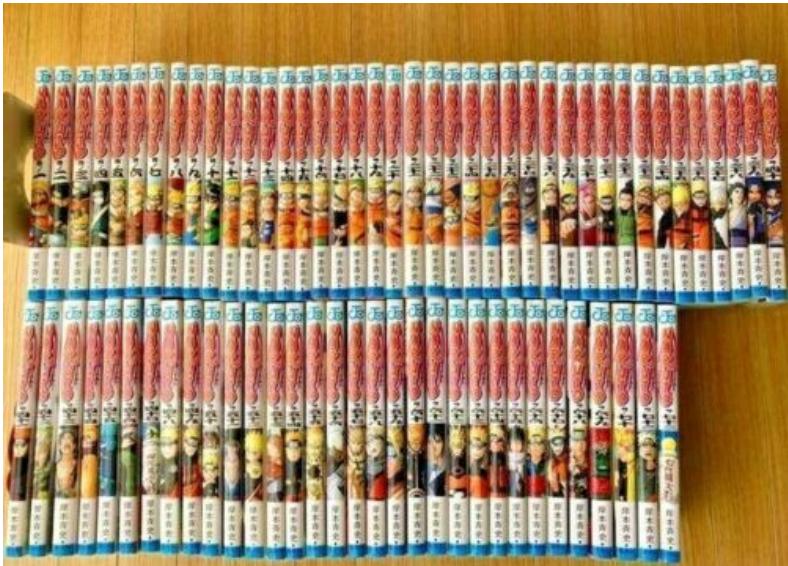
GPT-3



Two Stacked Eren Founding Titan 始祖の巨人  $2 \times 1000\text{m}$ <sup>40</sup>

# What is the scale 570GB

Read Naruto **270K** times



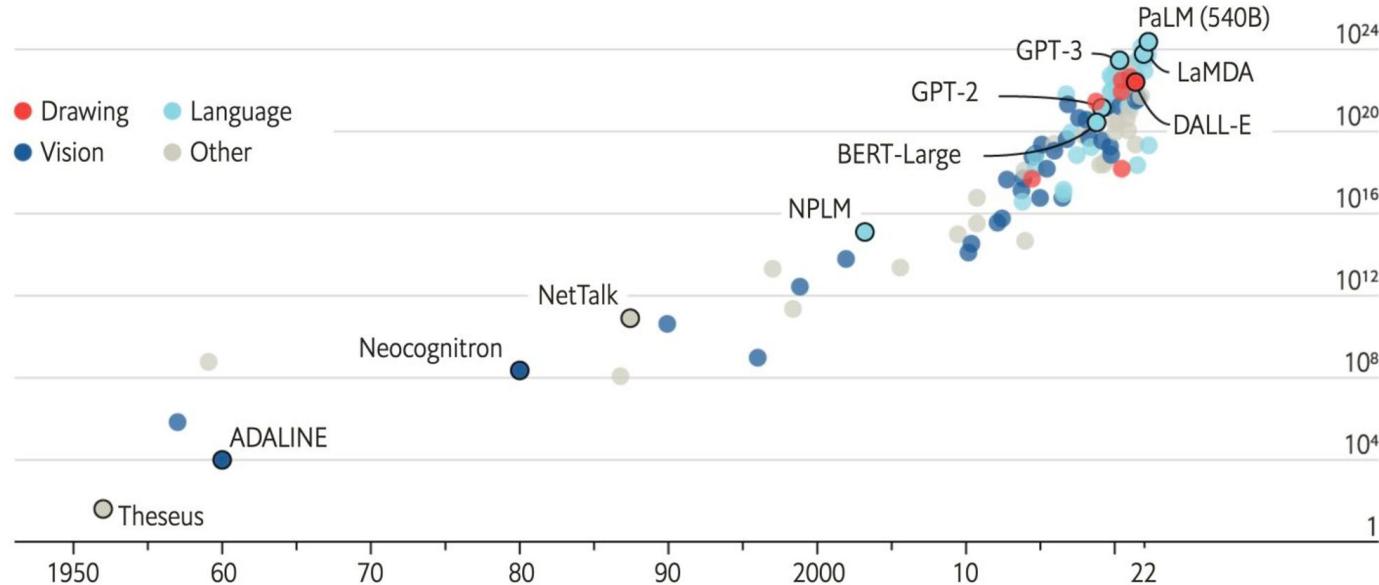
Only lines/text are counted.

# Pre-training

## The blessings of scale

AI training runs, estimated computing resources used

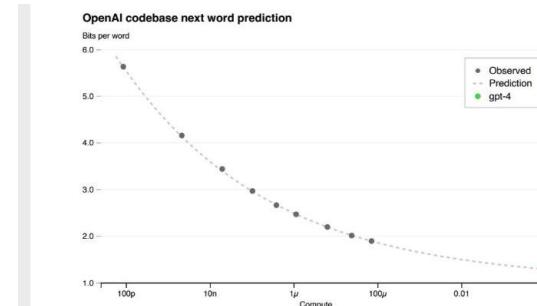
Floating-point operations, selected systems, by type, log scale



Sources: "Compute trends across three eras of machine learning", by J. Sevilla et al., arXiv, 2022; Our World in Data

# Predict Performances of LLM

- Since training massive models requires significant investment, we need approaches to predict performance before committing resources
- 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**

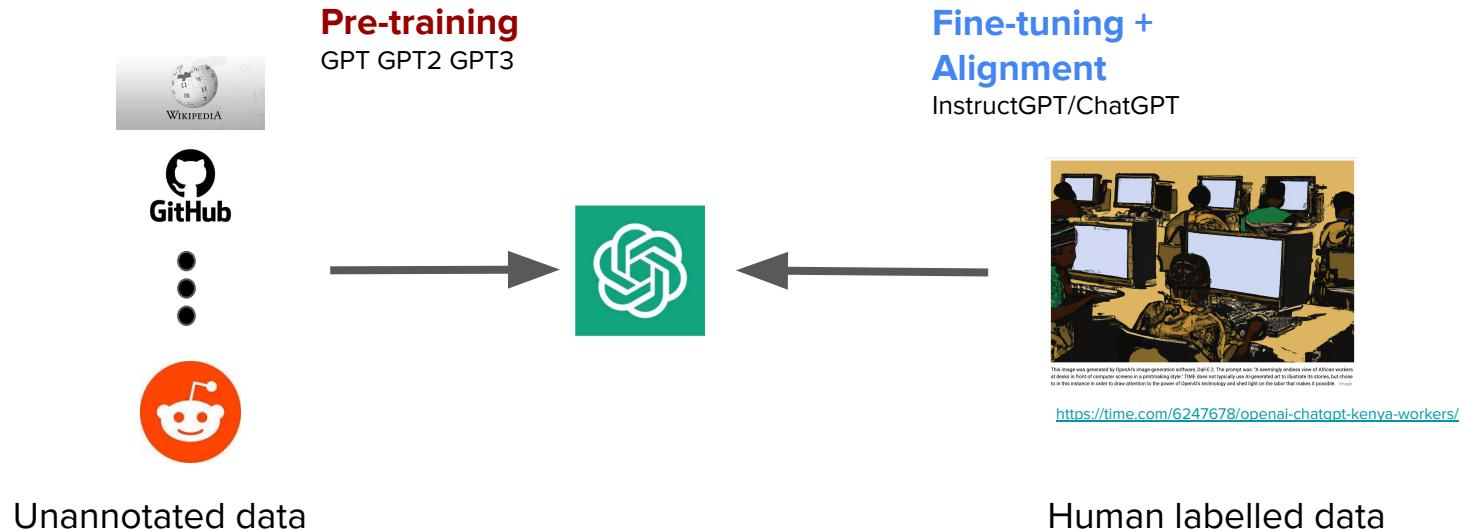
2:06 pm · 14 Nov 2024 · 2.5M Views

# 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

## 5. From GPT3 to ChatGPT: Finetuning + Alignment

# How to train ChatGPT



# GPT3 is already powerful

- GPT3 can do everything that Chatgpt can do
  - <https://pub.towardsai.net/crazy-gpt-3-use-cases-232c22142044>
- Why not achieve the hype as ChatGPT?
  - It is a language model that is learned from all text across the open internet
  - GPT3 is a genius but do not know our humans' preference

# GPT3's problem

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]
```



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]
```



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

# GPT3's problem

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]
```



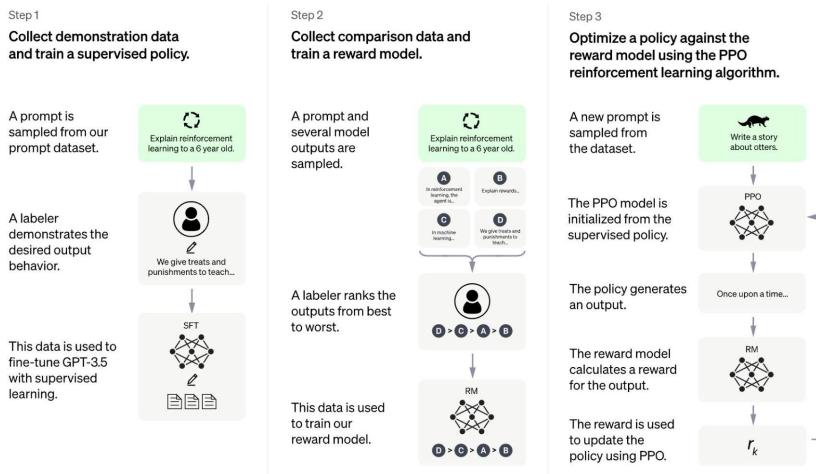
- A. to store the value of C[0]
- B. to store the value of C[1]
- C. to store the value of C[i]
- D. to store the value of C[i - 1]



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

# How to inject human preference

- This is how ChatGPT is different from GPT3.
- Reinforcement Learning from Human Feedback (RLHF):



## 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

# ChatGPT: Finetuning + Alignment

- It is not fully disclosed
- GPT3.5 +
  - SFT
  - RLHF
- Data quality and data collection is one key factor
  - From GPT3, Open AI has a framework to upweight high quality training text, and filter low quality ones
- Rumor:
  - About 10 times spend on human annotation budget
  - Improved version of RLHF

**In the following, we will focus on 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



**Try to increase the diversity and coverage of all prompts**

# Step 1: Supervised fine-tuning (InstructGPT)

- 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

It is too **expensive** to prepare the prompts and answer pair in the above way. Sometime, it is even not possible. Think about some prompts such as “write a song for my best friend who is having the kaggle competition”

# Step 1: Supervised fine-tuning (InstructGPT)

- 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

It is too **expensive** to prepare prompts in the above way. Sometime, it is even not possible. Think about some prompts such as “write a song for my best friend who is having the kaggle competition”

**We can not write songs but we can pick a better song.**

**Do you prefer MCQ or SAQ?**

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

# Step 2: 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]

# Step 3: Reinforcement learning (InstructGPT)

- Prepare PPO dataset (31k prompts)
  - Selected prompts and RM model from Step 2
- Fine-tune SFT model using PPO algorithm (one of Reinforcement Learning algorithms)
  - The environment here is: prompt is randomly presented and the SFT model is expected to generate a prompt
  - Given the prompt and response, RM model will output a reward value
  - SFT model is trained to achieve a higher reward value

**From these three steps, we kind of enforce GPT model to learn our humans' preference**

# How good is RLHF

1.3B RLHF outperform 175B GPT3 on human preferences

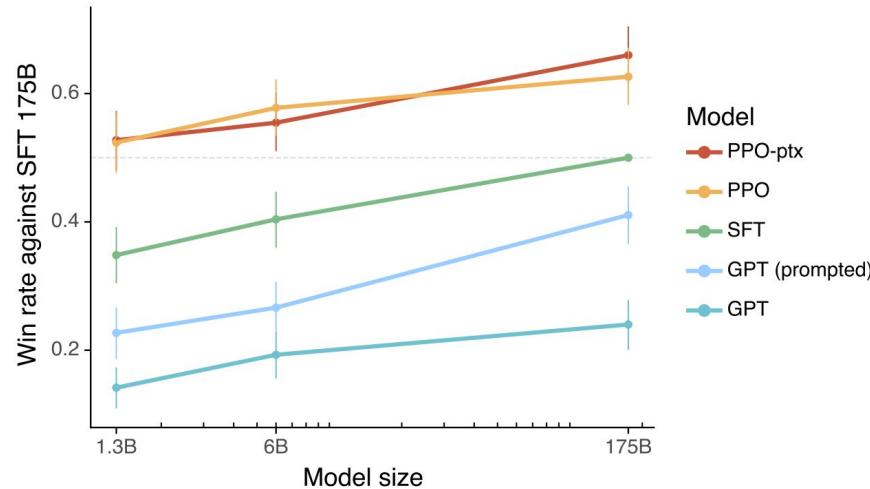


Figure 1: Human evaluations of various models on the API prompt distribution, evaluated by how often outputs from each model were preferred to those from the 175B SFT model. Our InstructGPT models (PPO-ptx) as well as its variant trained without pretraining mix (PPO) significantly outperform the GPT-3 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 ChatGPT
  - 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: Finetuning + 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

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

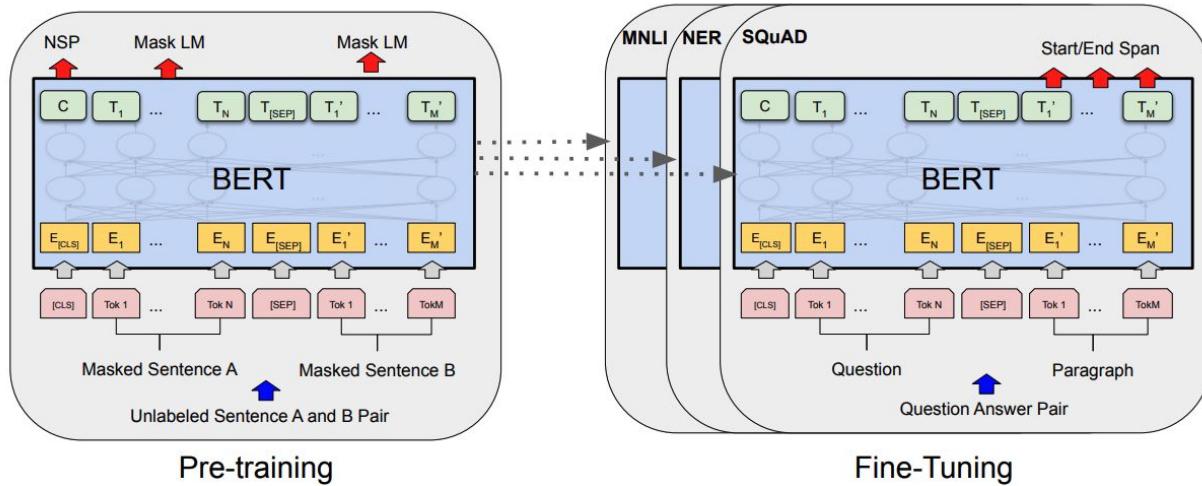


Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are 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

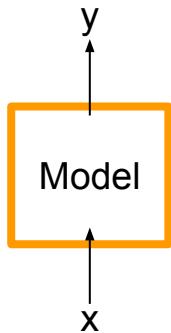
2019年4月30日 · 🌎

...

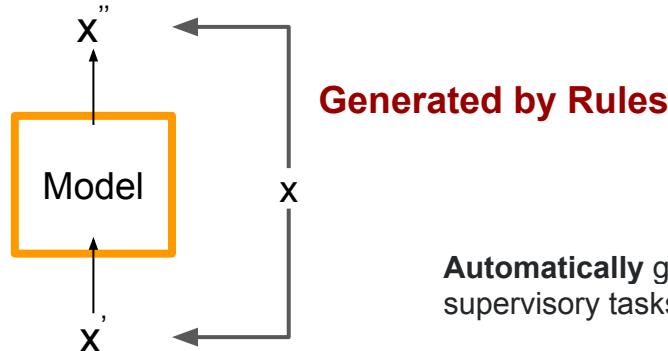
The answer is **self-supervised learning**.

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



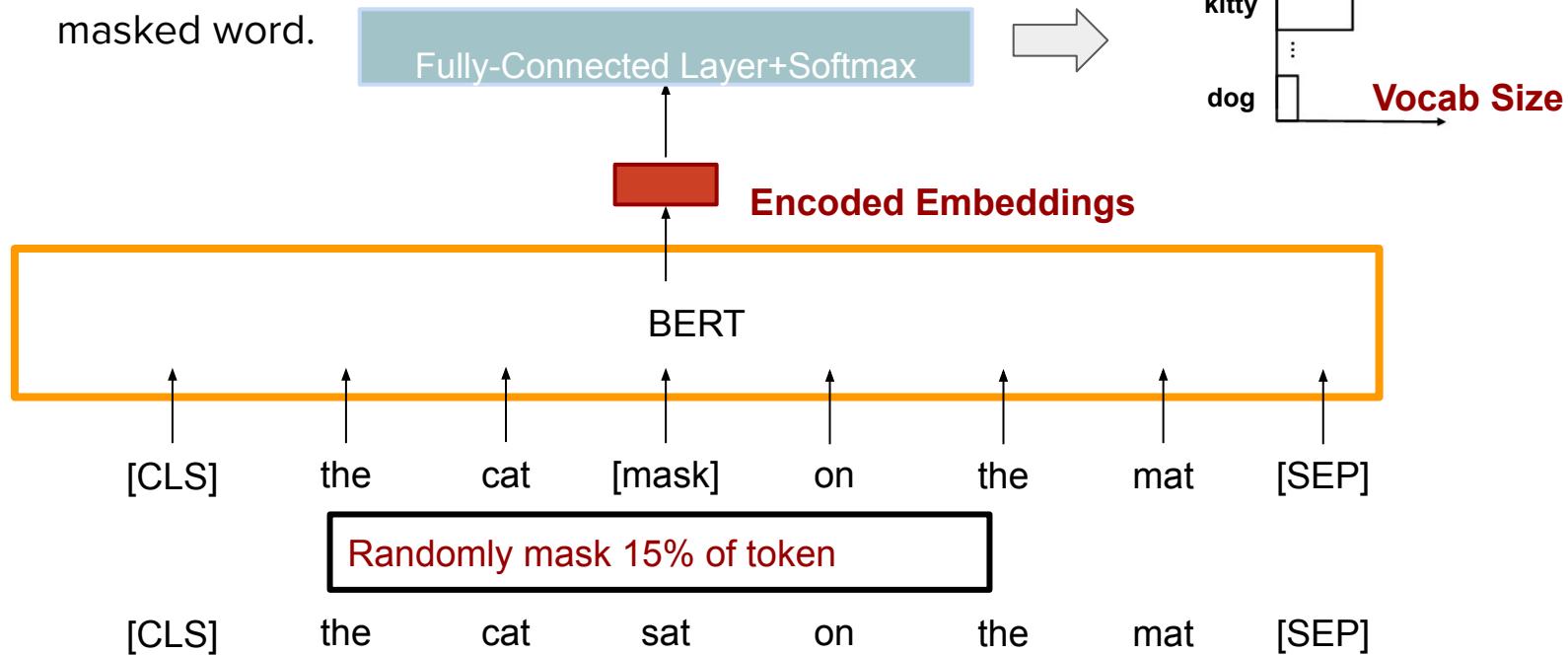
Self-Supervised

Automatically generate some kind of supervisory tasks

# Pre-training Task I: MLM

- **Masked Language Model**

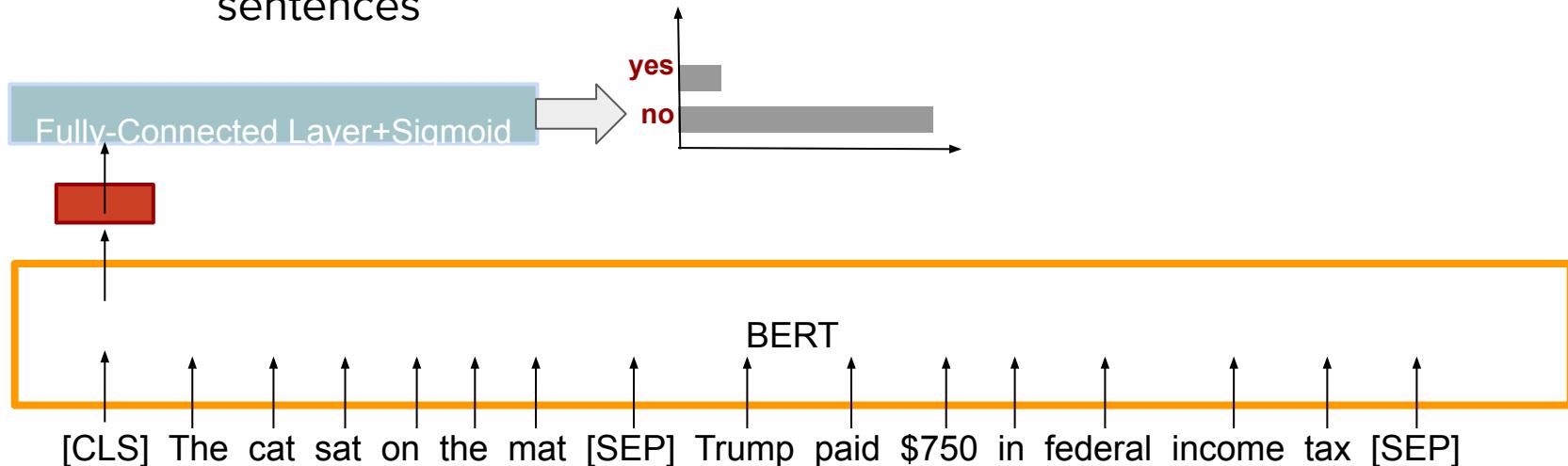
- Use the encoded embeddings of the masked word's to predict the masked word.



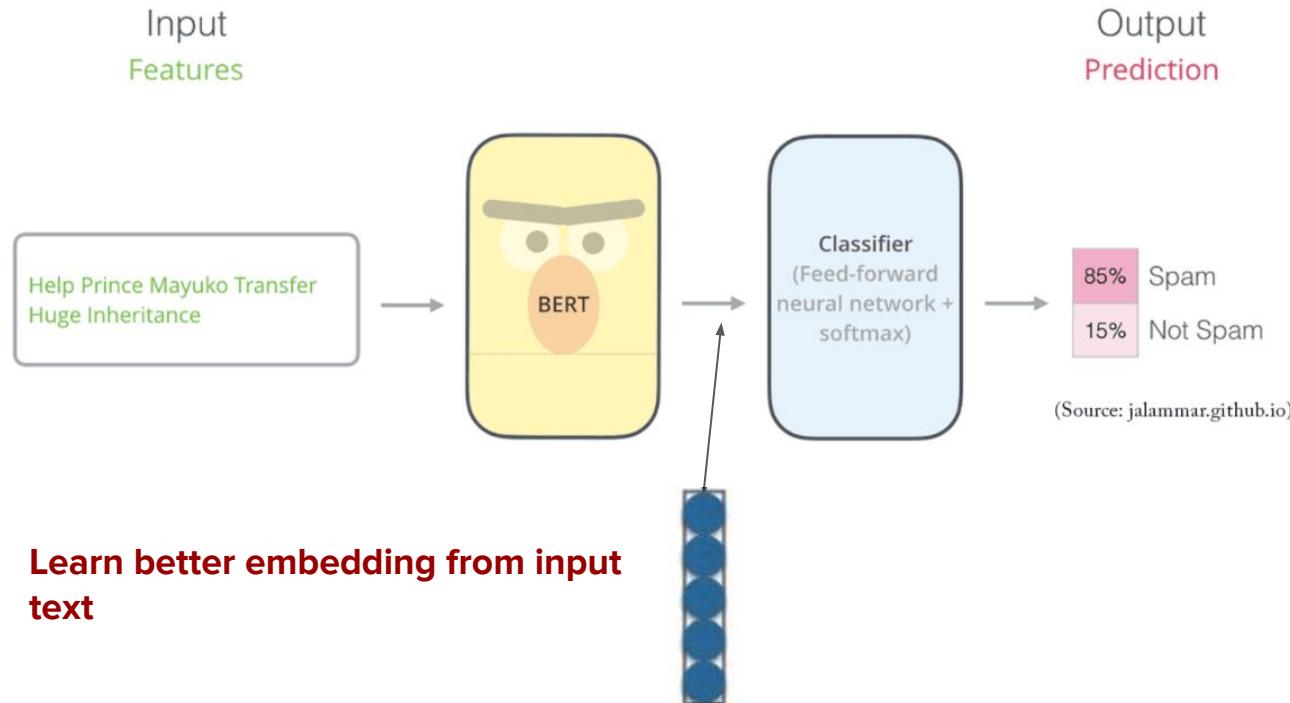
# Pre-training Task II: NSP

- **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



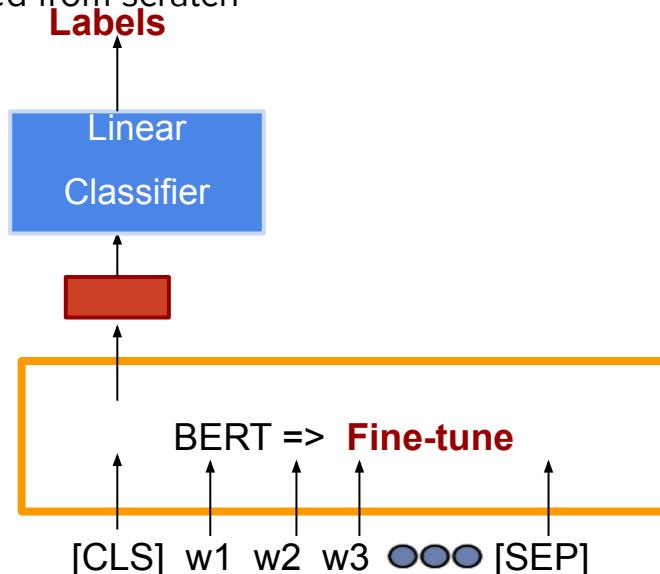
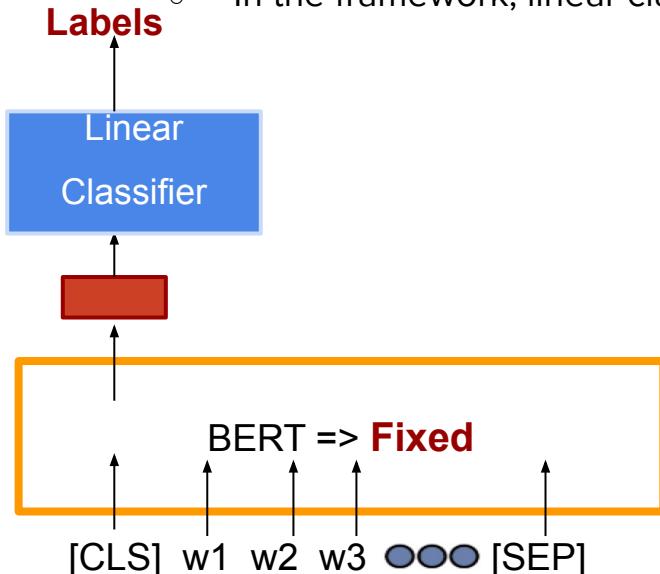
# Bert + Fine-tuning



# 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

