Language Models are Few-Shot Learners

Shaozhi Wang

Table of Content

Background

Hacker News
new | threads | past | comments | ask | show | jobs | submit
wporr (39) | logout

1. Feeling unproductive? Maybe you should stop overthinking (adolos.substack.com)
47 points by adolos 1 hour ago | flag | hide | 26 comments
2. ▲ 'Doomscrolling' Breeds Anxiety. Here's How to Stop the Cycle (npr.org)
34 points by mrfusion 1 hour ago | flag | hide | 24 comments
3. ▲ Why OKRs might not work at your company (svpg.com)
136 points by codesuki 4 hours ago | flag | hide | 49 comments

Nothing but Words

Feeling unproductive? Maybe you should stop overthinking.

LIAM PORR
2020年7月19日

36 43 Share

Background

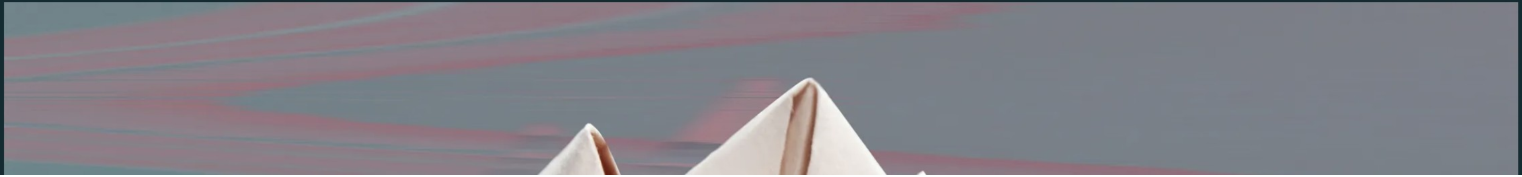
ARTIFICIAL INTELLIGENCE

A college kid's fake, AI-generated blog fooled tens of thousands. This is how he made it.

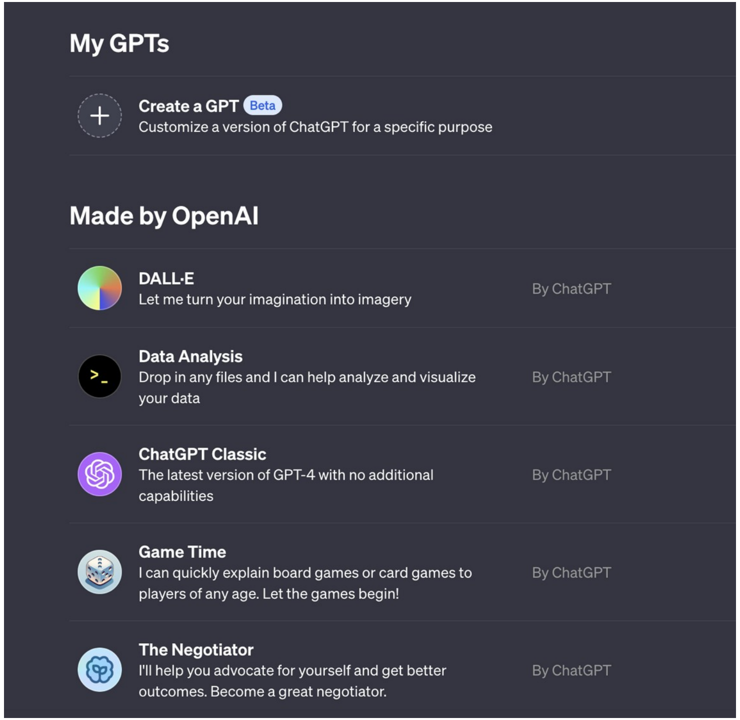
"It was super easy actually," he says, "which was the scary part."

By Karen Hao

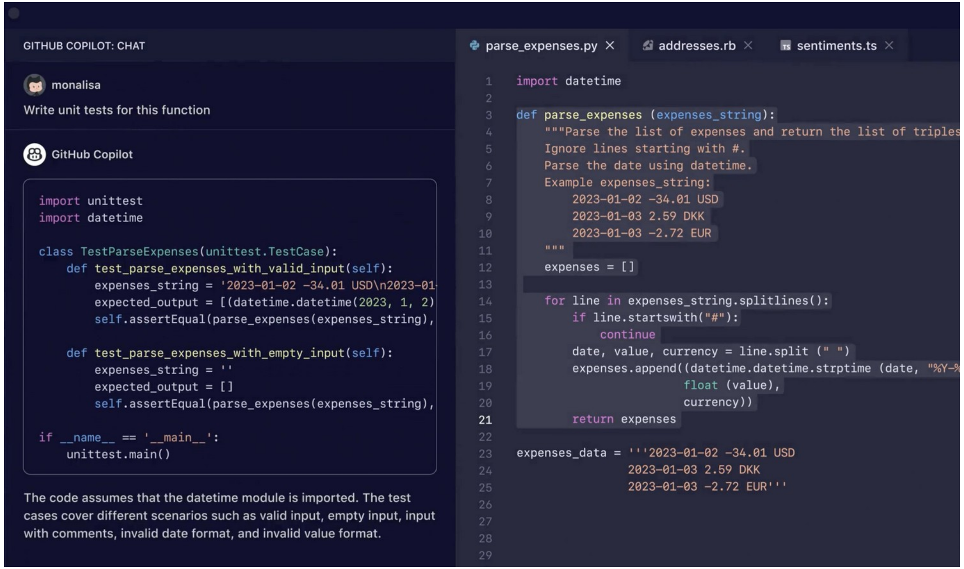
August 14, 2020



Background

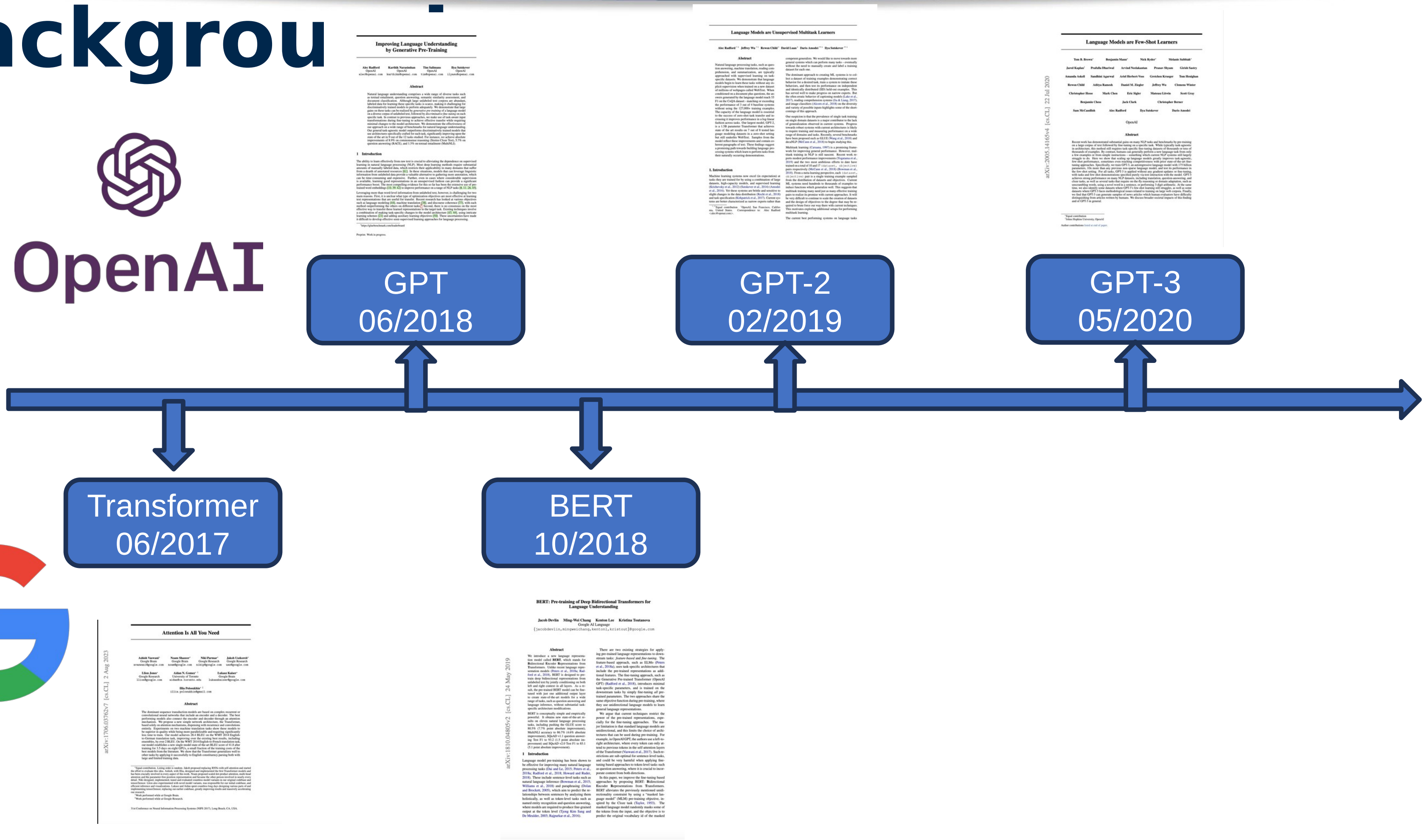


My GPTs
Create a GPT Beta
Customize a version of ChatGPT for a specific purpose
Made by OpenAI
DALL·E
Let me turn your imagination into imagery
By ChatGPT
Data Analysis
Drop in any files and I can help analyze and visualize your data
By ChatGPT
ChatGPT Classic
The latest version of GPT-4 with no additional capabilities
By ChatGPT
Game Time
I can quickly explain board games or card games to players of any age. Let the games begin!
By ChatGPT
The Negotiator
I'll help you advocate for yourself and get better outcomes. Become a great negotiator.
By ChatGPT



GITHUB COPILOT: CHAT
parse_expenses.py
addresses.rb
sentiments.ts
monalisa
Write unit tests for this function
GitHub Copilot
import unittest
import datetime
class TestParseExpenses(unittest.TestCase):
def test_parse_expenses_with_valid_input(self):
def test_parse_expenses_with_empty_input(self):
if __name__ == '__main__':
unittest.main()
The code assumes that the datetime module is imported. The test cases cover different scenarios such as valid input, empty input, input with comments, invalid date format, and invalid value format.
import datetime
def parse_expenses (expenses_string):
Ignore lines starting with #.
Parse the date using datetime.
Example expenses_string:
2023-01-02 -34.01 USD
2023-01-03 2.59 DKK
2023-01-03 -2.72 EUR
expenses = []
for line in expenses_string.splitlines():
if line.startswith("#"):
continue
date, value, currency = line.split (" ")
return expenses
expenses_data = '''2023-01-02 -34.01 USD
2023-01-03 2.59 DKK
2023-01-03 -2.72 EUR'''

Backgrou



Improving Language Understanding by Generative Pre-Training
Language Models are Unsupervised Multitask Learners
Language Models are Few-Shot Learners
GPT
06/2018
GPT-2
02/2019
GPT-3
05/2020
Transformer
06/2017
BERT
10/2018
Attention Is All You Need
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Related Work: GPT-1

- **Improving Language Understanding by Generative Pre-Training**
  - Natural language understanding comprises a wide range of diverse tasks, although large unlabeled text corpora are abundant, labeled data for learning specific tasks is scarce, making it challenging for discriminatively trained models to perform adequately.
  - **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, they make use of **task-aware input transformations** during fine-tuning to achieve effective transfer while **requiring minimal changes** to the model architecture.

Related Work: GPT-1

- **Introduction**

  - In NLP, most deep learning methods require substantial amounts of manually **labeled data**, which restricts their applicability in many domains
  - In these situations, models that can leverage linguistic information from **unlabeled data** provide a valuable alternative

  - However, two main challenges of these models:
    - ➢ It is unclear what type of optimization objectives are most effective at learning text representations (Depends on different tasks)
    - ➢ There is no consensus on the most effective way to transfer these learned representations to the target tasks (Since the target tasks are quite different with each other)

Related Work: GPT-1

- **Introduction**
  - For model architecture, they use Transformer
  - Compared with RNN, transformer provides with a more structured memory for handling long-term dependencies in text, resulting in robust transfer performance across diverse tasks

Related Work: GPT-1

- **Framework**
  - Unsupervised pre-training:

Given an unsupervised corpus of tokens $\mathcal{U} = \{u_1, \ldots, u_n\}$
They use a standard language modeling objective to maximize the following likelihood:

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta)$$

  - k is the size of the context window, and the conditional probability P is modeled using a neural network with parameters Θ.

Related Work: GPT-1

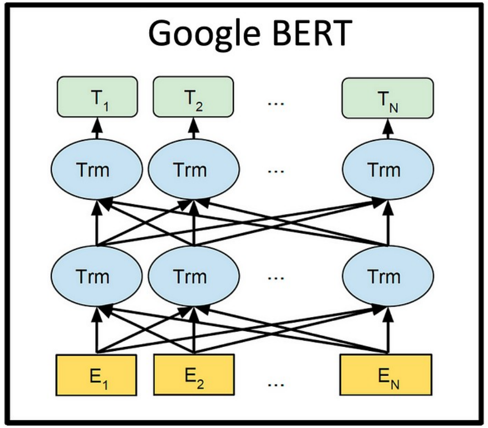
- **Framework**
  - Unsupervised pre-training:
  - Use a multi-layer **Transformer decoder** for the language model, which is a variant of the transformer. This model applies a multi-headed self-attention operation over the input context tokens followed by position-wise feedforward layers to produce an output distribution over target tokens:

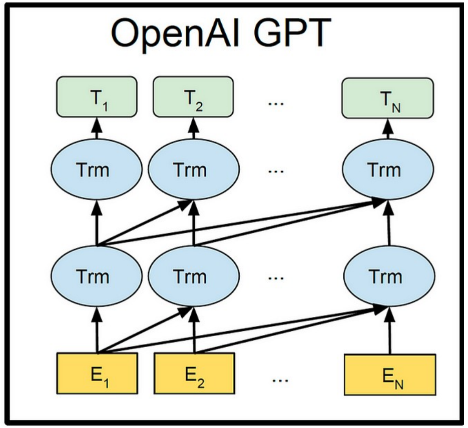
$$
\begin{aligned}
h_0 &= UW_e + W_p \\
h_l &= \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n] \\
P(u) &= \texttt{softmax}(h_n W_e^T)
\end{aligned}
$$

  - Where. $U = (u_{-k}, \ldots, u_{-1})$ is the context vector of tokens, ***n*** is the number of layers, $\boldsymbol{W_e}$ is the token embedding matrix, and $\boldsymbol{W_p}$ is the position embedding matrix.

Related Work: BERT

- **Framework**
  - GPT versus BERT: Decoder vs Encoder: (Attention masks)
  - Different objective functions: prediction vs cloze
  - Also, BERT's training dataset is larger than GPT's training dataset:
  - The above is why BERT's performance is better than GPT





Related Work: GPT-2

- **Language Models are Unsupervised Multitask Learners**

  - Compared with GPT-1 and BERT:
  - ➢ Larger training dataset: trained on a new dataset of millions of webpages.
  - ➢ Larger models: 1.5B parameter Transformer (BERT-large, 0.34 B parameter)

  - ➢ Zero-shot setting:
  - **In GPT-1: Generative Pre-Training** of a language model on a diverse corpus of **unlabeled text**, followed by **discriminative fine-tuning** on **each specific task.**
  - **In GPT-2: without further fine-tuning on down-stream tasks, highlighting the Generalizability**

  - However, Innovative but lacks effectiveness

GPT-3:

- **Language Models are Few-Shot Learners**

  - Give-up GPT-2’s zero-shot learning, instead, using **Few-shot** learning:

  - Different with GPT-1, 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
  - ➢ Why?
  - ➢ Large model with 175 billion parameters!

  - GPT-3 can generate samples of news articles which human evaluators have difficulty distinguishing from articles written by humans.

GPT-3:

- **Introduction:**
  - Recent years have featured a trend towards **pre-trained language representations** in NLP system, and task-agnostic ways for downstream transfer (Just like GPT-1)
  - However, limitations: there is still a need for **task-specific datasets** and **task-specific fine-tuning**: to achieve strong performance.
  - Removing the above limitation would be desirable, for several reasons:

  - ➢ First, from a practical perspective, the need for a large dataset of labeled examples for every new task limits the applicability of language models.
  - ➢ Second, the potential to exploit spurious correlations in training data fundamentally grows with the expressiveness of the model and the narrowness of the training distribution.
  - ➢ Third, humans do not require large supervised datasets to learn most language tasks

GPT-3:

- **Introduction:**

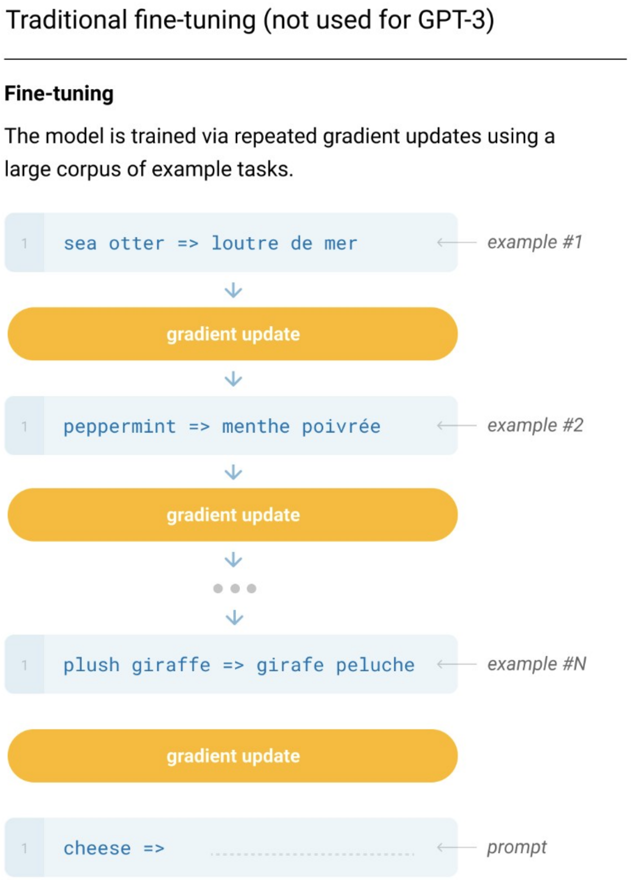
Solution: “Meta-learning” and “In-context learning”:

- **Meta-learning**: they trained a large and generalizable model
- The model develops a broad set of skills and pattern recognition abilities at training time, and then uses those abilities at inference time to rapidly adapt to or recognize the desired task.

- **In-context learning**: No updates will be made to the model weights
- Using the text input of a pretrained language model as a form of task specification:

GPT-3: Approach

- **Methods:**
  - Training a 175 billion parameter autoregressive language model
  - Fine-tuning:



GPT-3: Approach

- **Methods:**
    - Zero-shot:
    - One-shot:

Zero-shot

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

```
1  Translate English to French:     ← task description
2  cheese => ...................    ← prompt
```

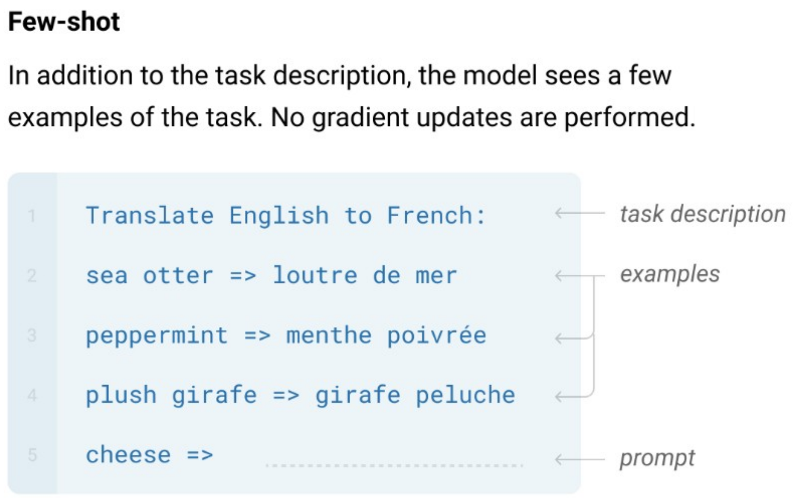
One-shot

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

```
1  Translate English to French:     ← task description
2  sea otter => loutre de mer       ← example
3  cheese => ...................    ← prompt
```

GPT-3: Approach

- **Methods:**
  - Few-shot:



GPT-3: Approach

- **Methods:**

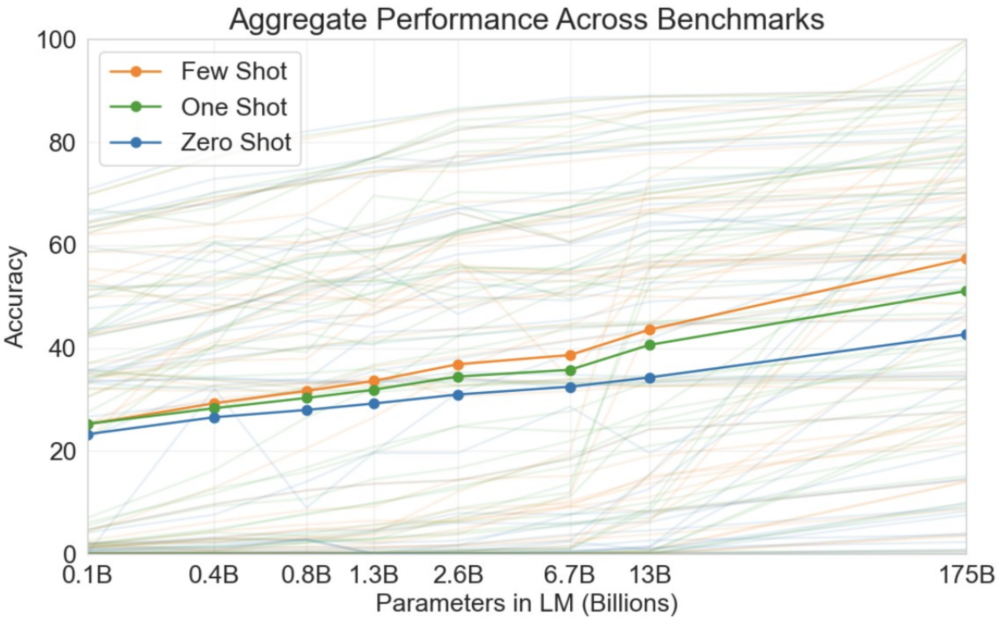


Figure 1.3: Aggregate performance for all 42 accuracy-denominated benchmarks While zero-shot performance improves steadily with model size, few-shot performance increases more rapidly, demonstrating that larger models are more proficient at in-context learning. See Figure 3.8 for a more detailed analysis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

GPT-3: Approach

- **Model and Architectures**
  - The same model and architecture as GPT-2:

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

Table 2.1: Sizes, architectures, and learning hyper-parameters (batch size in tokens and learning rate) of the models which we trained. All models were trained for a total of 300 billion tokens.

GPT-3: Approach

- **Training Dataset:**
  - Filtered a version of CommonCrawl based on similarity to a range of high-quality reference corpora
  - Performed fuzzy deduplication at the document level, within and across datasets, to prevent redundancy and preserve the integrity of their held-out validation set as an accurate measure of overfitting (LSH algorithms)
  - Added known high-quality reference corpora to the training mix to augment CommonCrawl and increase its diversity.

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

Table 2.2: Datasets used to train GPT-3. "Weight in training mix" refers to the fraction of examples during training that are drawn from a given dataset, which we intentionally do not make proportional to the size of the dataset. As a result, when we train for 300 billion tokens, some datasets are seen up to 3.4 times during training while other datasets are seen less than once.

GPT-3: Approach

- **Training Process:**
  - All models were trained on V100 GPU’s on part of a high-bandwidth cluster provided by Microsoft
  - Not open to public: 175 billion parameters

GPT-3: Evaluation

- **Evaluation Process:**

  - For few-shot learning, they evaluate each example in the evaluation set by randomly drawing K examples from that task's training set as conditioning, delimited by 1 or 2 newlines depending on the tasks
  - K can be any value from 0 to the maximum amount allowed by the model's context window, typically fits **10** to **100** examples.

➢ On tasks that involve **multiple choice**:
They provide K examples of context plus correct completion, followed by one example of context only, and compare the LM likelihood of each completion.
➢ On tasks that involve **binary classification**:
They give the options more semantically meaningful names (e.g. "True" or "False" rather than 0 or 1) and then treat the task like multiple choice
➢ On tasks with **free-form completion**, they use beam search
Beam width of 4 and a length penalty of $\alpha = 0.6$. They score the model using F1 similarity score, BLEU, or exact match, depending on what is standard for the dataset at hand.

GPT-3: Evaluation

- **Result:**

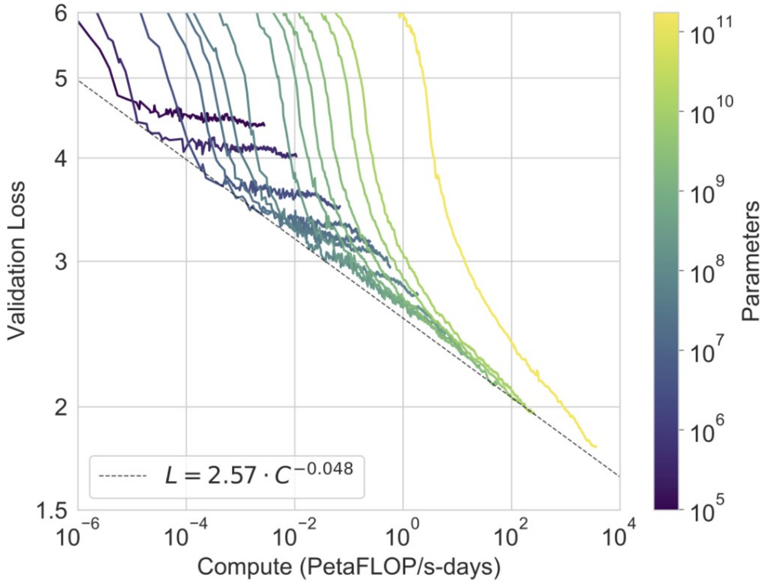


Figure 3.1: Smooth scaling of performance with compute. Performance (measured in terms of cross-entropy validation loss) follows a power-law trend with the amount of compute used for training. The power-law behavior observed in [KMH+20] continues for an additional two orders of magnitude with only small deviations from the predicted curve. For this figure, we exclude embedding parameters from compute and parameter counts.

GPT-3: Evaluation

- **Result:**

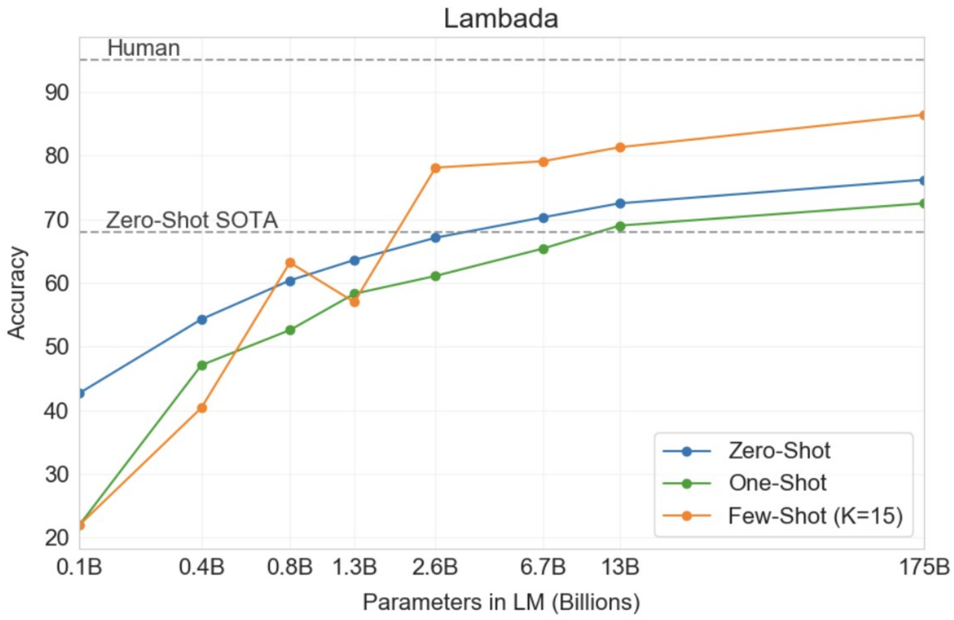


Figure 3.2: On LAMBADA, the few-shot capability of language models results in a strong boost to accuracy. GPT-3 2.7B outperforms the SOTA 17B parameter Turing-NLG [Tur20] in this setting, and GPT-3 175B advances the state of the art by 18%. Note zero-shot uses a different format from one-shot and few-shot as described in the text.

GPT-3: Evaluation

- **Result:**

Setting	NaturalQS	WebQS	TriviaQA
RAG (Fine-tuned, Open-Domain) [LPP+20]	**44.5**	**45.5**	**68.0**
T5-11B+SSM (Fine-tuned, Closed-Book) [RRS20]	36.6	44.7	60.5
T5-11B (Fine-tuned, Closed-Book)	34.5	37.4	50.1
GPT-3 Zero-Shot	14.6	14.4	64.3
GPT-3 One-Shot	23.0	25.3	**68.0**
GPT-3 Few-Shot	29.9	41.5	**71.2**

Table 3.3: Results on three Open-Domain QA tasks. GPT-3 is shown in the few-, one-, and zero-shot settings, as compared to prior SOTA results for closed book and open domain settings. TriviaQA few-shot result is evaluated on the wiki split test server.

GPT-3: Evaluation

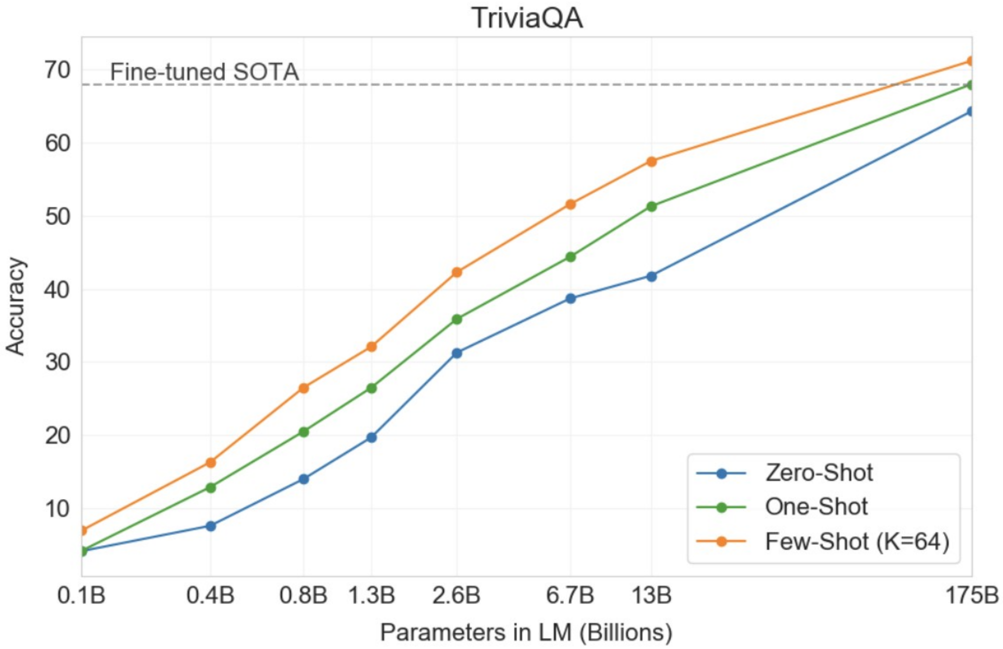


Figure 3.3: On TriviaQA GPT3's performance grows smoothly with model size, suggesting that language models continue to absorb knowledge as their capacity increases. One-shot and few-shot performance make significant gains over zero-shot behavior, matching and exceeding the performance of the SOTA fine-tuned open-domain model, RAG [LPP+20]

GPT-3: Evaluation

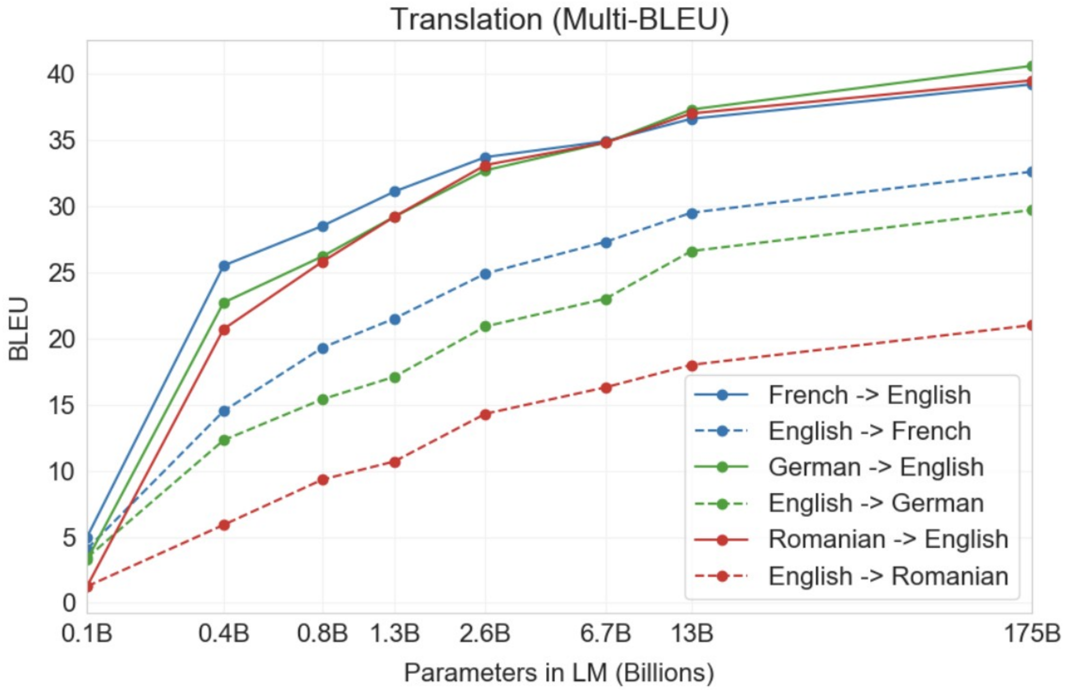


Figure 3.4: Few-shot translation performance on 6 language pairs as model capacity increases. There is a consistent trend of improvement across all datasets as the model scales, and as well as tendency for translation into English to be stronger than translation from English.

GPT-3: Limitation

- **Limitation:**
  - Still has notable weaknesses in text synthesis and several NLP tasks.
  - Has several structural and algorithmic limitations
  - Objective weights every token equally and lacks a notion of what is most important to predict and what is less important
  - Not grounded in other domains of experience, such as video or real-world physical interaction
  - Uncertainty associated with few-shot learning in GPT-3 is ambiguity about whether few-shot learning actually learns new tasks “from scratch” at inference time, or if it simply recognizes and identifies tasks that it has learned during training
  - Expensive
  - Its decisions are not easily interpretable (unexplainable)

GPT-3: Quiz

-
    - What is the main difference between GPTs and BERT in Architecture?

GPT-3: Quiz

-
  - What is the main difference between GPTs and BERT in Architecture?

  - GPT-1: It is an autoregressive model, which means it predicts the next word in a sentence by considering only the words that came before it.
  - BERT: Contrary to GPT-1, BERT is a bidirectional model. It is designed to understand the context of a word based on all surrounding words (both before and after the word), not just the words that precede it.

GPT-3: Quiz

-
  - In what ways does GPT-3 improve upon the limitations identified in GPT-2?

    A. Increased Scale

    B. Better Context Understanding

    C. Enhanced Learning from Zero-shot to Few-shot

    D. All of the above

GPT-3: Quiz

-
  - In what ways does GPT-3 improve upon the limitations identified in GPT-2?

A. Increased Scale

B. Better Context Understanding

C. Enhanced Learning from Zero-shot to Few-shot

D. All of the above

GPT-3: Discussion

Q & A