Annotated Data

Working with LLMs: Prompting

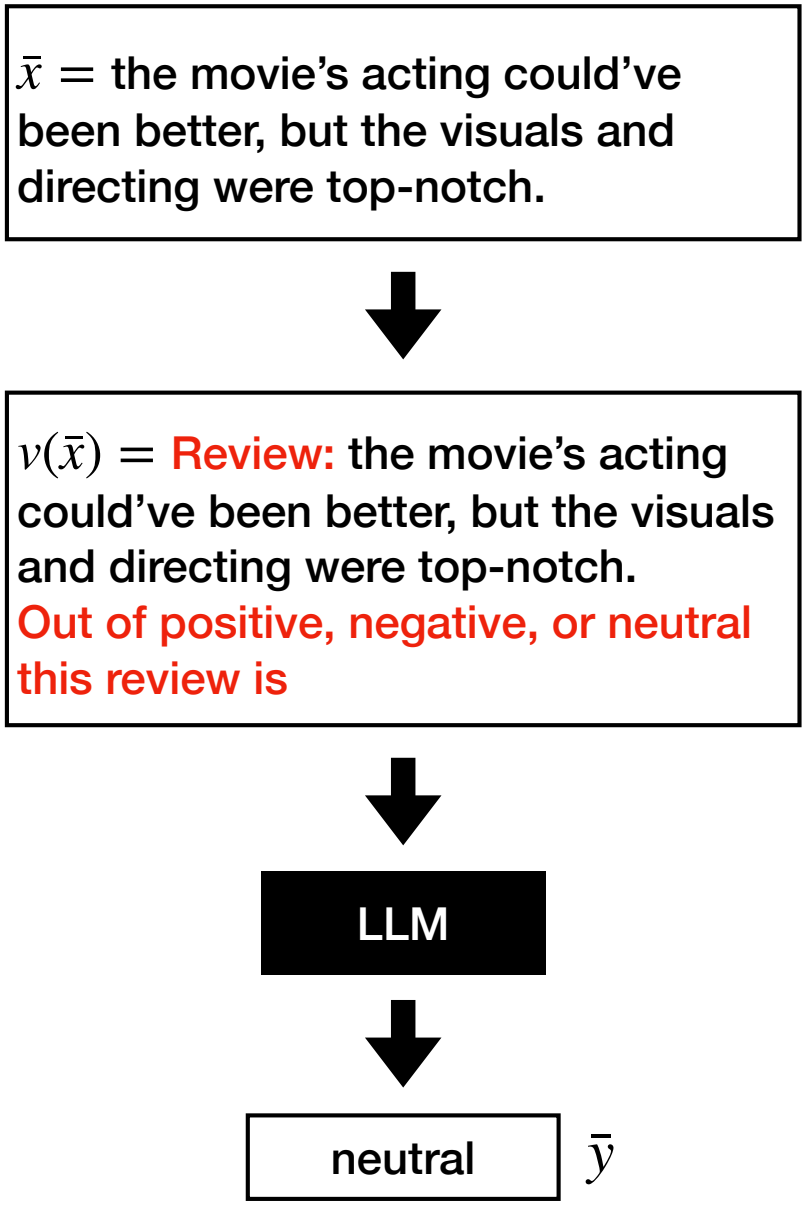
Cornell CS 5740: Natural Language Processing
Yoav Artzi, Spring 2023

Working with LLMs

- A simple way to turn LLMs into task-specific models is through fine-tuning
  - Identical to what we saw with BERT: fine-tune with annotated data
  - You benefit from the rich representations of the LLM
- LLMs offer a completely new mode of operation that does not require any change to their parameters: **prompting**
  - With or without annotated examples: ***zero-shot*** or ***in-context learning*** (few-shot)
  - With or without intermediate reasoning steps: ***chain-of-thought*** prompting

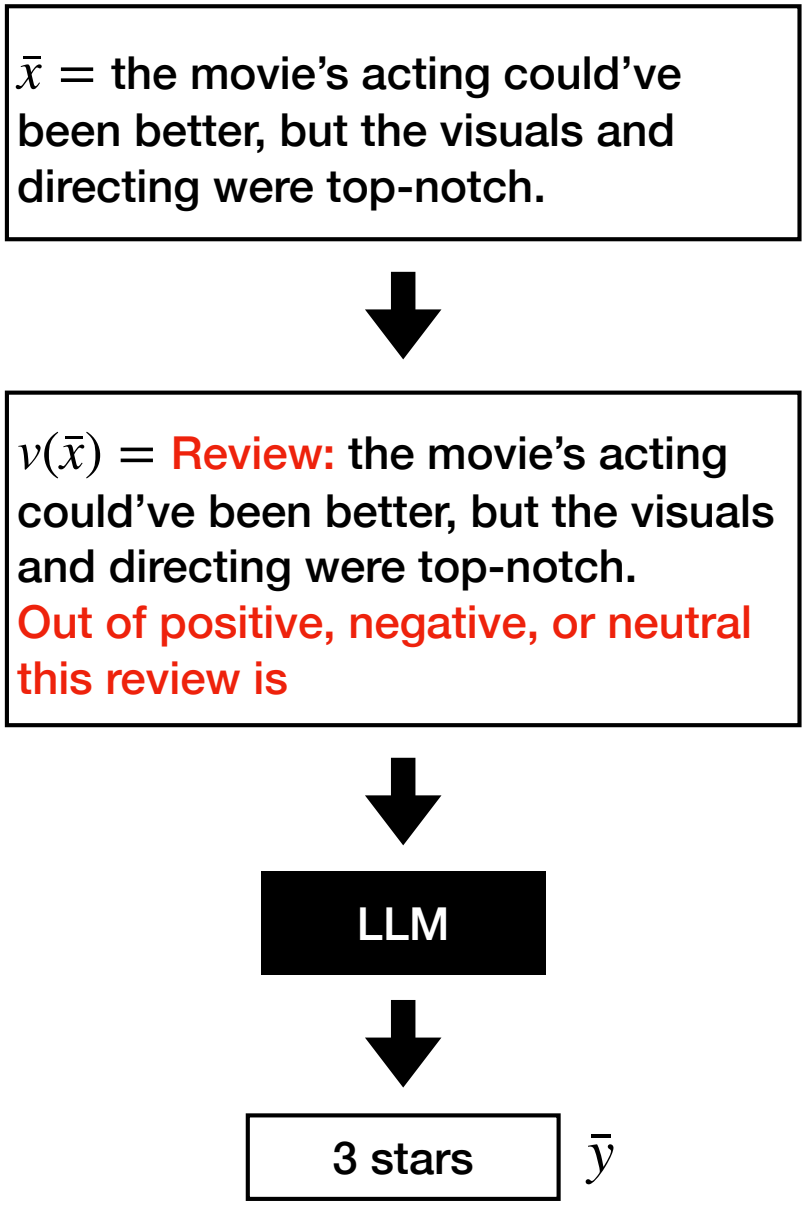
Zero-shot Prompting

- Input: single unlabeled example $\bar{x}$
- Output: the label $\bar{y}$
- The task (and output) can be any text-to-text task: classification, summarization, translation
- Pre-processing: wrap $\bar{x}$ with a template using a ***verbalizer*** v
- The template controls the output



Zero-shot Prompting

- Input: single unlabeled example $\bar{x}$
- Output: the label $\bar{y}$
- The task (and output) can be any text-to-text task: classification, summarization, translation
- Pre-processing: wrap $\bar{x}$ with a template using a ***verbalizer*** v
- The template controls the output



Zero-shot Prompting

Constrained Output

- We generate from the model to get the output
  - What if the model output does not fit the intended format, even if it is semantically correct?
    - "… how many stars on a scale of four? 4" vs.
      "… how many stars on a scale of four? four stars"
  - Or maybe not even semantically correct, but just irrelevant?

Zero-shot Prompting

Constrained Output

- We generate from the model to get the output
  - What if the model output does not fit the intended format, even if it is semantically correct?
    - "… how many stars on a scale of four? 4" vs.
      "… how many stars on a scale of four? four stars"
- Generate with constraints:
  - Compare the probabilities of all possible outputs according to your format

$$\arg\max_{\bar{y}\in\{1,2,3,4\}} p(\bar{y} \mid v(\bar{x}))$$

Zero-shot Prompting

Constrained Output

- Generate with constraints:
  - Compare the probabilities of all possible outputs according to your format

$$\arg\max_{\bar{y} \in \{1,2,3,4\}} p(\bar{y} \mid v(\bar{x}))$$

  - If the label is a single token ($|\bar{y}| = 1$), just compare next token probabilities over labels
  - Otherwise?

Zero-shot Prompting

Constrained Output

- Generate with constraints:
  - Compare the probabilities of all possible outputs according to your format

$$\arg\max_{\bar{y} \in \{1,2,3,4\}} p(\bar{y} \mid v(\bar{x}))$$

  - If the label is a single token ($|\bar{y}| = 1$), just compare next token probabilities over labels
  - Otherwise, compute $p(\bar{y} \mid v(\bar{x}))$ by force decoding the considered output (why? can we avoid this?)
  - Can normalize to get a distribution between only valid outputs

Zero-shot Prompting

Constrained Output

- Generate with constraints:
  - Compare the probabilities of all possible outputs according to your format

$$\arg\max_{\bar{y} \in \{1,2,3,4\}} p(\bar{y} \mid v(\bar{x}))$$

  - If the label is a single token ($|\bar{y}| = 1$), just compare next token probabilities over labels
  - Otherwise, compute $p(\bar{y} \mid v(\bar{x}))$ by force decoding the considered output (why? can we avoid this?)
  - Can normalize to get a distribution between only valid outputs
  - When is this hard?

Zero-shot Prompting

Sensitivity and Variability

- Prompting simplifies some aspects of adapting LLMs for tasks
  - No need to do expensive parameter estimate
  - You need much less data: no training data with zero-shot prompting
- However: many sources of unexpected variability
  - There are many way to write a prompt for the same task
  - Can we expect all of them to simply function the same?

Zero-shot Prompting

Sensitivity and Variability

- Prompts create a natural language input
- So the model ability to reason about that language influences task performance
  - How “natural” it is?
  - How does it “align” with the training data?

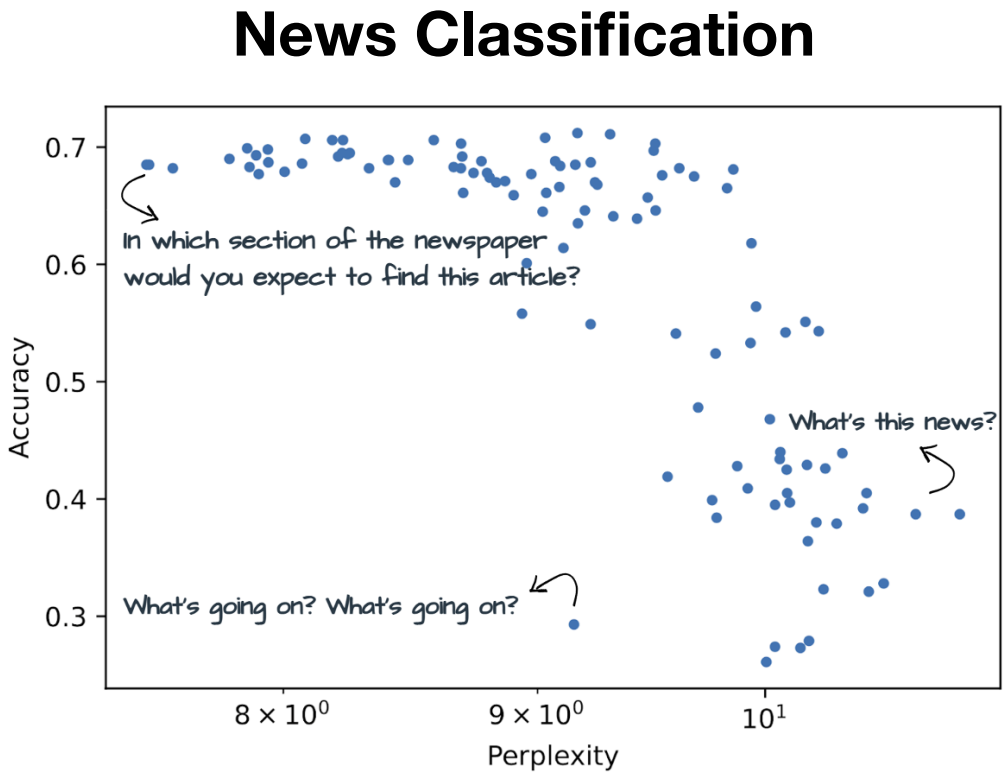


Figure 1: Accuracy vs. perplexity for the AG News dataset with OPT 175B. The x axis is in log scale. Each point stands for a different prompt.

[Gonen et al. 2022]

Zero-shot Prompting

Sensitivity and Variability

- Minor changes that should have no impact, can have dramatic effect
- For example: asking for answer in quotations

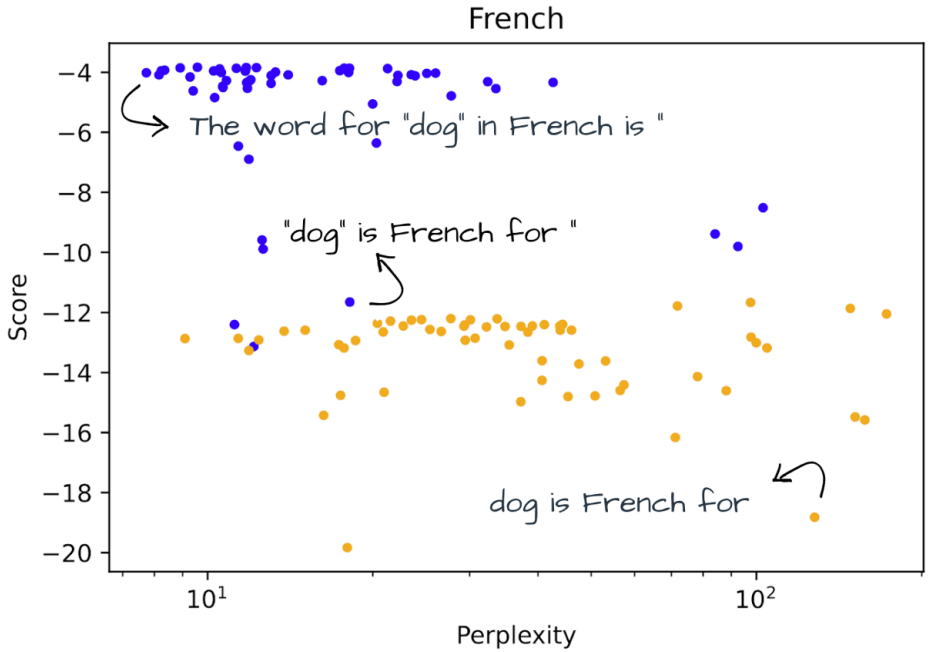


Figure 2: Score of correct label vs. perplexity for the word-level translation task in French with OPT 175B. The x axis is in log scale. The blue points stand for prompts with quotation marks for the words, while the yellow points are of prompts without quotation marks.

[Gonen et al. 2022]

Zero-shot Prompting

Sensitivity and Variability

- Across open-weight models (at the time), mean best 50% of prompts perform much better than all prompts
- Caveat: it is an open question how this generalizes to current state-of-the-art models

Model	Task	Avg Acc	Acc 50%
OPT 175B	Antonyms	–	–
	GLUE Cola	47.7	57.1
	Newspop	66.4	72.9
	AG News	57.5	68.7
	IMDB	86.2	91.0
	DBpedia	46.7	55.2
	Emotion	16.4	23.0
	Tweet Offensive	51.3	55.8
Bloom 176B	Antonyms	–	–
	GLUE Cola	55.5	65.6
	Newspop	78.9	87.8
	AG News	50.3	59.4
	IMDB	89.3	92.2
	DBpedia	27.2	33.4
	Emotion	29.3	31.7
	Tweet Offensive	41.6	46.2
OPT 30B	Antonyms	–	–
	GLUE Cola	32.2	35.5
	Newspop	60.3	66.6
	AG News	49.3	60.7
	IMDB	81.6	86.1
	DBpedia	35.9	42.4
	Emotion	12.3	16.2
	Tweet Offensive	54.6	60.2
OPT 1.3B	Antonyms	–	–
	GLUE Cola	60.3	65.9
	Newspop	37.6	40.3
	AG News	31.9	37.6
	IMDB	86.0	89.1
	DBpedia	8.7	9.2
	Emotion	7.0	9.1
	Tweet Offensive	58.6	62.6

[Gonen et al. 2022]

Zero-shot Prompting

Sensitivity and Variability

- Prompts can even be sensitive to minor cosmetic changes
- Can influence performance in unexpected ways
- Can think of them as (very complex) hyper-parameters

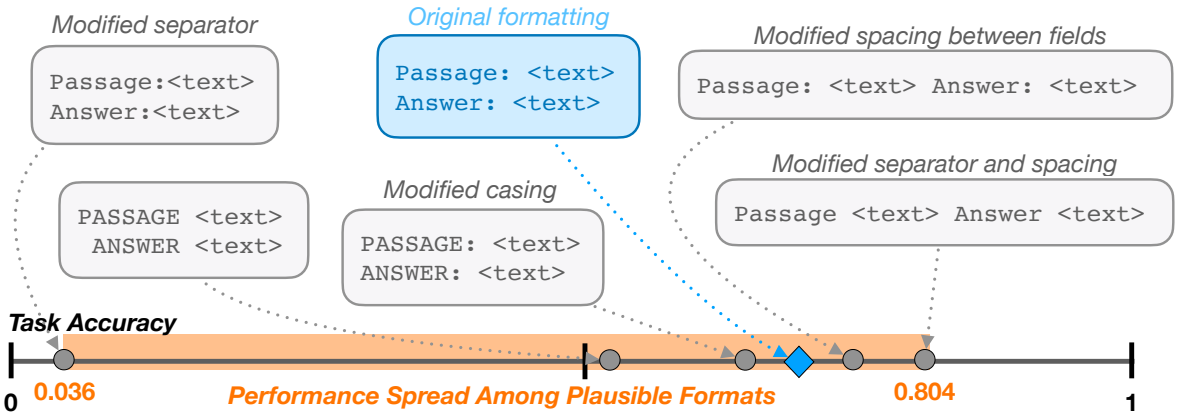


Figure 1: Slight modifications in prompt format templating may lead to significantly different model performance for a given task. Each `<text>` represents a different variable-length placeholder to be replaced with actual data samples. Example shown corresponds to 1-shot LLaMA-2-7B performances for task280 from SuperNaturalInstructions (Wang et al., 2022). This StereoSet-inspired task (Nadeem et al., 2021) requires the model to, given a short passage, classify it into one of four types of stereotype or anti-stereotype (gender, profession, race, and religion).

[Sclar et al. 2023]

Zero-shot Prompting

Sensitivity and Variability

- Prompts can even be sensitive to minor cosmetic changes
- Can influence performance in unexpected ways
- Can think of them as (very complex) hyper-parameters

Task Id	Prompt Format 1 (p_1)	Prompt Format 2 (p_2)	Acc p_1	Acc p_2	Diff.
task280	`passage:{}\n answer:{}`	`passage {}\n answer {}`	0.043	0.826	0.783
task317	`Passage::{} Answer::{}`	`Passage:: {} Answer:: {}`	0.076	0.638	0.562
task190	`Sentence[I]- {}Sentence[II]- {} -- Answer\t{}`	`Sentence[A]- {}Sentence[B]- {} -- Answer\t{}`	0.360	0.614	0.254
task904	`input:: {} \n output:: {}`	`input::{} \n output::{}`	0.418	0.616	0.198
task320	`target - {} \n{} \nanswer - {}`	`target - {}; \n{}; \nanswer - {}`	0.361	0.476	0.115
task322	`COMMENT: {} ANSWER: {}`	`comment: {} answer: {}`	0.614	0.714	0.100
task279	`Passage : {}. Answer : {}`	`PASSAGE : {}. ANSWER : {}`	0.372	0.441	0.069

[Sclar et al. 2023]

Zero-shot Prompting

Surface Form Competition

- Given a closed set of answers, humans can explicitly restrict their choice
- Even if you constrain a model, the entire vocabulary is competing
- A very similar answer might get suck probability from the right one, but still be considered wrong

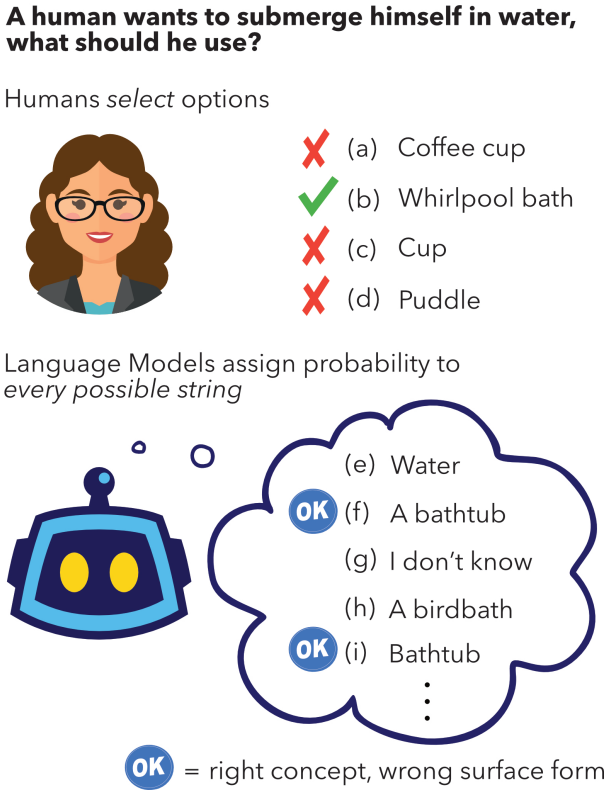


Figure 1: While humans select from given options, language models implicitly assign probability to every possible string. This creates surface form competition between different strings that represent the same concept. Example from CommonsenseQA (Talmor et al., 2019).

[Holtzman et al. 2021]

Zero-shot Prompting

Prompt Optimization

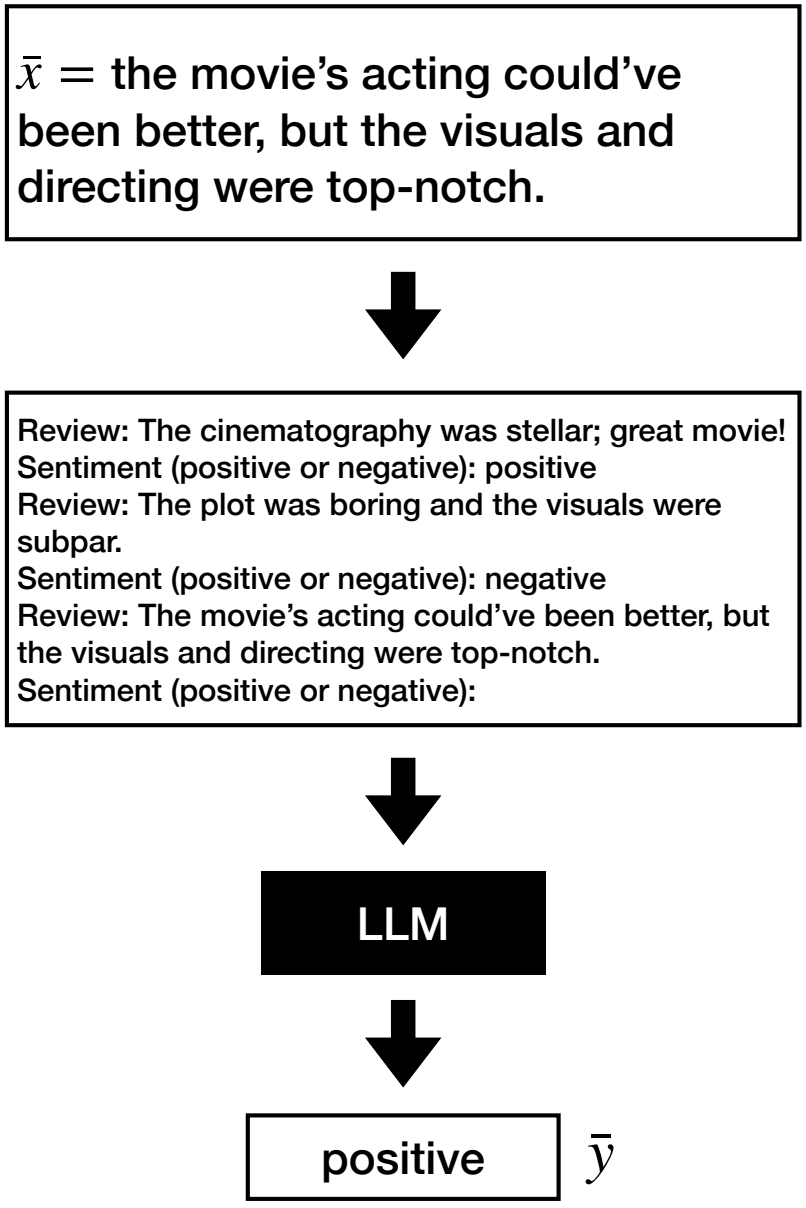
- Just like hyper-parameters, can think of optimizing prompts
- There are methods for searching over prompts (either using gradients or black-box optimization)
- Most do not lead to dramatically better results compared to manual engineering/hill-climbing (and are computationally intensive)
- Most important: the choice of prompt is very important for zero-shot settings

In-context Learning (ICL)

- LLMs have the ability to “learn” to complete tasks through training in the prompt
- The recipe is simple:
  - Take a small number of annotated training example $\{(\bar{x}^{(i)}, \bar{y}^{(i)}\}_{i=1}^{N}$
  - Convert them using verbalizer v templates
  - Concatenate them and follow with the target input $\bar{x}$
  - The completion will be the label of the input

In-context Learning (ICL)

- LLMs have the ability to "learn" to complete tasks through training in the prompt
- The recipe is simple:
  - Take a small number of annotated training example $\{(\bar{x}^{(i)}, \bar{y}^{(i)}\}_{i=1}^{N}$
  - Convert them using verbalizer v templates
  - Concatenate them and follow with the target input $\bar{x}$
  - The completion will be the label of the input



In-context Learning (ICL)

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
```

Few-shot

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

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← examples
3  peppermint => menthe poivrée        ←
4  plush girafe => girafe peluche      ←
5  cheese =>                           ← prompt
```

Fine-tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.

```
1  sea otter => loutre de mer          ← example #1
```

gradient update

```
1  peppermint => menthe poivrée        ← example #2
```

gradient update

...

```
1  plush giraffe => girafe peluche     ← example #N
```

gradient update

```
1  cheese =>                           ← prompt
```


[Figure from Brown et al. 2020]

In-context Learning (ICL)

Performance

- Providing ICL examples almost always leads to significant improvements

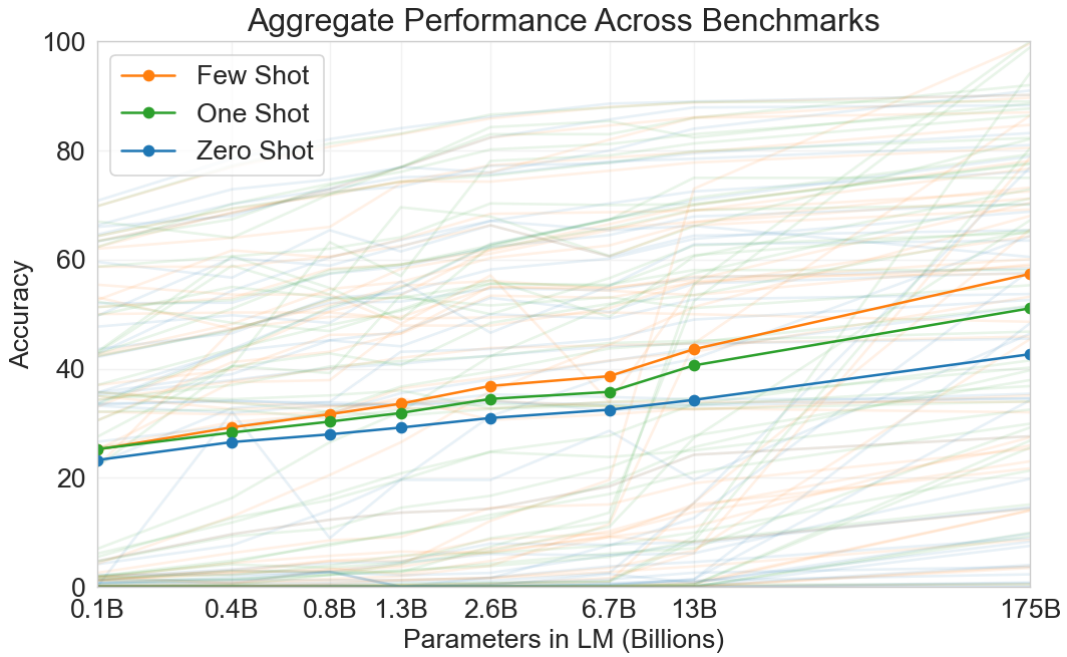


Figure 1.3: Aggregate performance for all 42 accuracy-denominated benchmarks While zero-shot performance improves steadily with model size, few-shot performance increases more rapidly, demonstrating that larger models are more proficient at in-context learning. See Figure 3.8 for a more detailed analysis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

[Brown et al. 2020]

In-context Learning (ICL)

Performance

- Providing ICL examples almost always leads to significant improvements
- Benefits tend to diminish with more examples

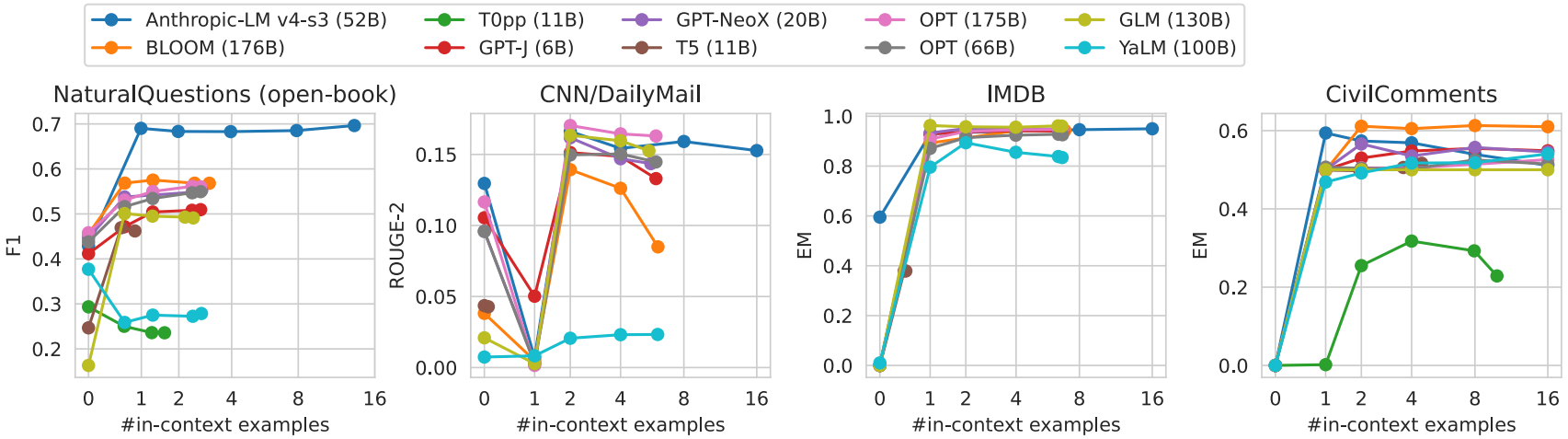


Figure 32: **Number of in-context examples.** For each model, we set the maximum number of in-context examples to [0, 1, 2, 4, 8, 16] and fit as many in-context examples as possible within the context window. We plot performance as a function of the average number of in-context examples actually used.

[Liang et al. 2022]

In-context Learning

Performance

- Model scale is important
- More examples have diminishing return
- What is the cost of more examples?

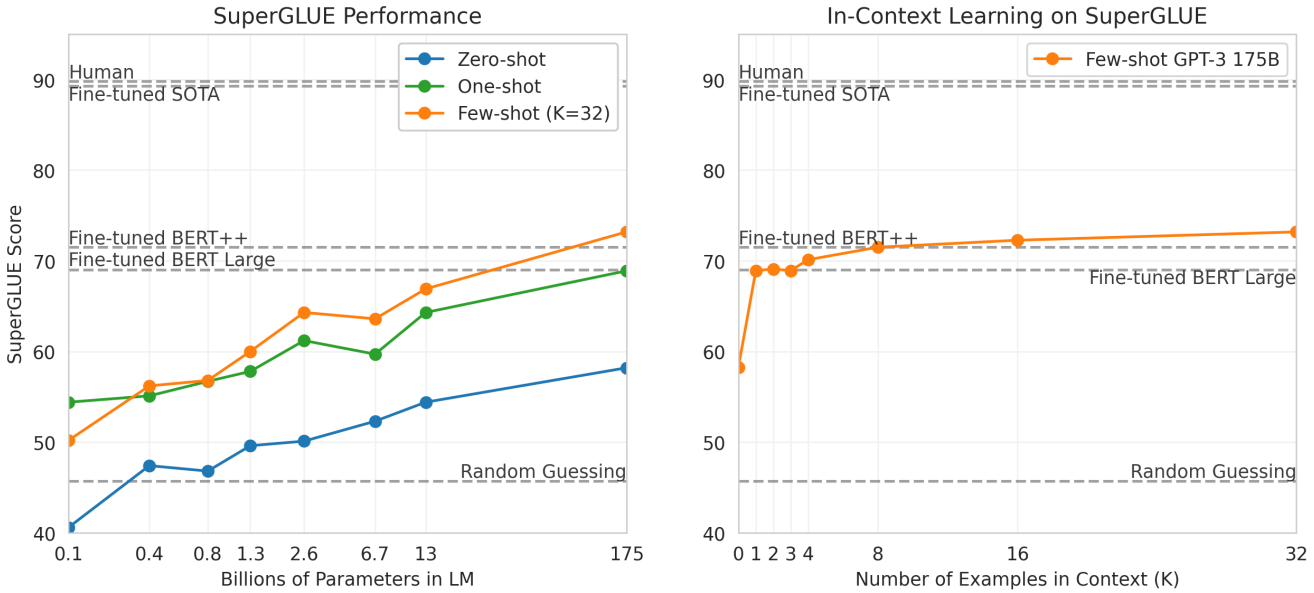


Figure 3.8: Performance on SuperGLUE increases with model size and number of examples in context. A value of $K = 32$ means that our model was shown 32 examples per task, for 256 examples total divided across the 8 tasks in SuperGLUE. We report GPT-3 values on the dev set, so our numbers are not directly comparable to the dotted reference lines (our test set results are in Table 3.8). The BERT-Large reference model was fine-tuned on the SuperGLUE training set (125K examples), whereas BERT++ was first fine-tuned on MultiNLI (392K examples) and SWAG (113K examples) before further fine-tuning on the SuperGLUE training set (for a total of 630K fine-tuning examples). We find the difference in performance between the BERT-Large and BERT++ to be roughly equivalent to the difference between GPT-3 with one example per context versus eight examples per context.

[Brown et al. 2020]

In-context Learning (ICL)

Sensitivity

- ICL can be highly sensitive to the choice of examples, their ordering, and the format of the prompt

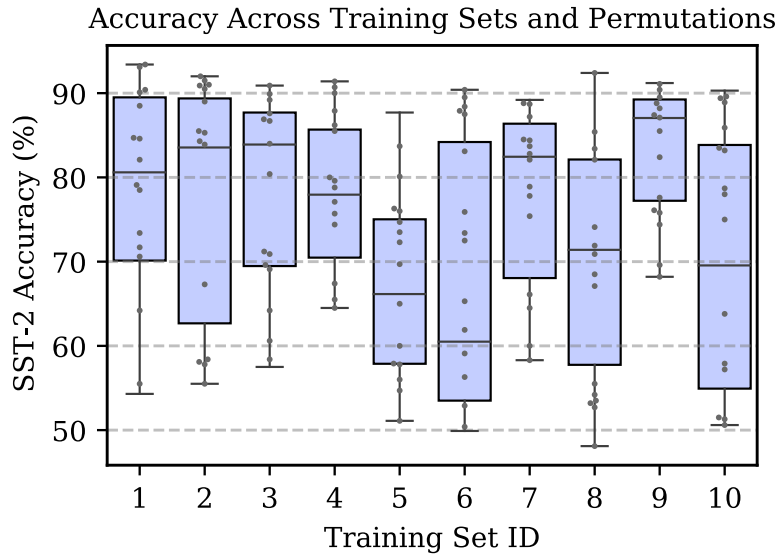


Figure 2. There is high variance in GPT-3's accuracy as we change the prompt's **training examples**, as well as the **permutation** of the examples. Here, we select ten different sets of four SST-2 training examples. For each set of examples, we vary their permutation and plot GPT-3 2.7B's accuracy for each permutation (and its quartiles).

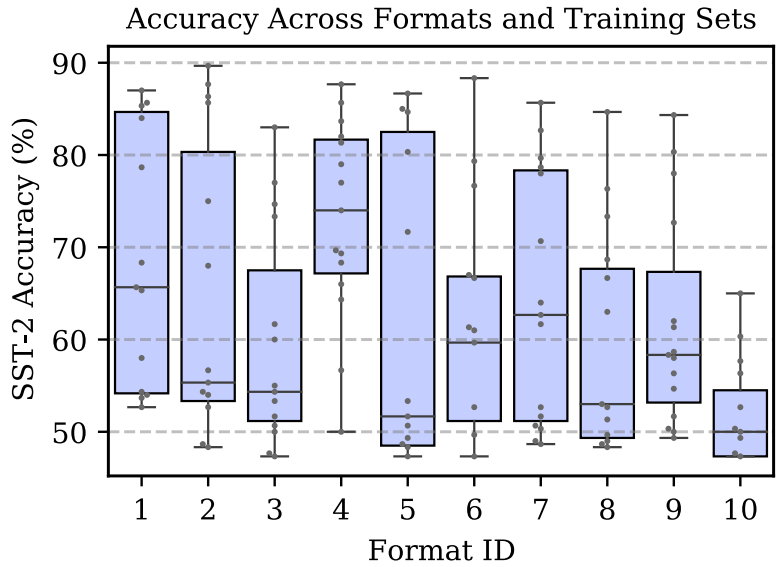


Figure 3. There is high variance in GPT-3's accuracy as we change the **prompt format**. In this figure, we use ten different prompt formats for SST-2. For each format, we plot GPT-3 2.7B's accuracy for different sets of four training examples, along with the quartiles.

[Zhao et al. 2021]

In-context Learning (ICL)

Sensitivity

- Ordering and choice of examples can lead to strong label bias

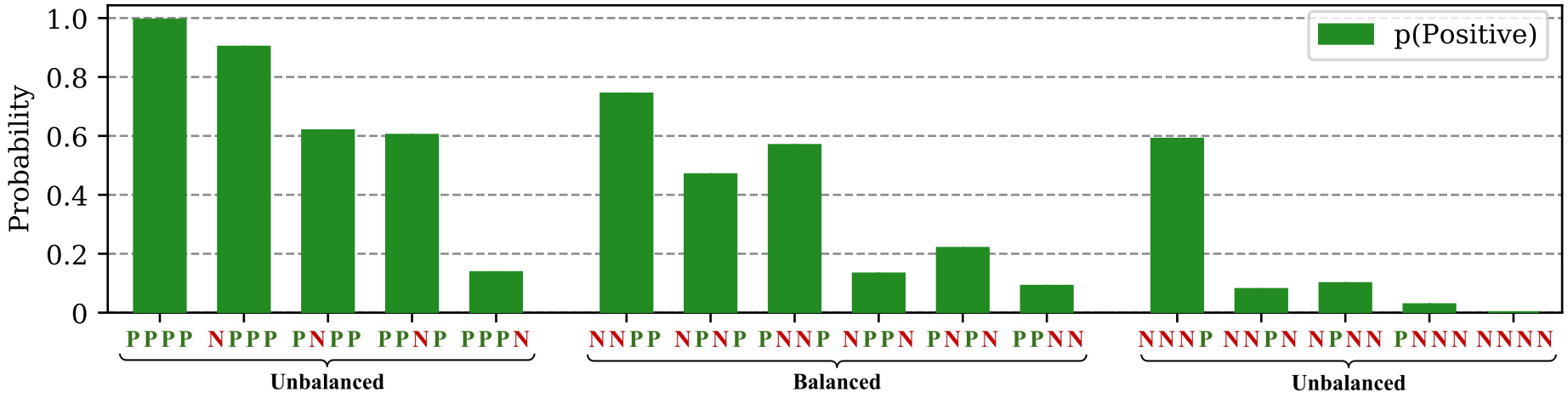


Figure 4. **Majority label and recency biases** cause GPT-3 to become biased towards certain answers and help to explain the high variance across different examples and orderings. Above, we use 4-shot SST-2 with prompts that have different class balances and permutations, e.g., [P P N N] indicates two positive training examples and then two negative. We plot how often GPT-3 2.7B predicts Positive on the balanced validation set. When the prompt is unbalanced, the predictions are unbalanced (*majority label bias*). In addition, balanced prompts that have one class repeated near the end, e.g., end with two Negative examples, will have a bias towards that class (*recency bias*).

[Zhao et al. 2021]

In-context Learning (ICL)

Sensitivity

- Particularly sensitive with fewer examples
  - Why using few examples is critical?
- There are methods that help, for examples see this tutorial

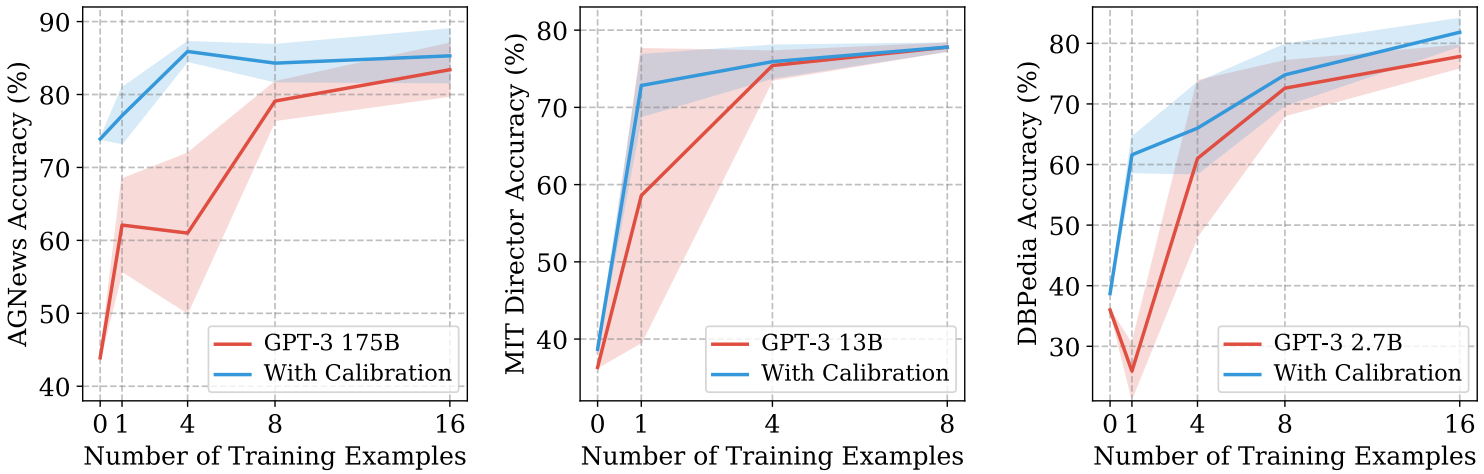


Figure 1. Few-shot learning can be highly unstable across different choices of the prompt. Above, we plot the mean accuracy (± one standard deviation) across different choices of the training examples for three different datasets and model sizes. We show that our method, *contextual calibration*, improves accuracy, reduces variance, and overall makes tools like GPT-3 more effective for end users.

[Zhao et al. 2021]

In-context Learning (ICL)

Analysis

- In some cases, the label correctness actually matters little
- But demonstrations still important
- What's happening? Demonstration are much about domain and form

Figure 1: Results in classification (top) and multi-choice tasks (bottom), using three LMs with varying size. Reported on six datasets on which GPT-3 is evaluated; the channel method is used. See Section 4 for the full results. In-context learning performance drops only marginally when labels in the demonstrations are replaced by random labels.

[Min et al. 2022]

Chain-of-thought (COT) Prompting

- Some tasks require multiple reasoning steps
- Directly generating the answer requires the model internally do the reasoning steps (or shortcut somehow)
- It is empirically useful to:
  - Show the model examples of the reasoning steps through ICL
  - And then have it explicitly generate the reasoning steps

[Wei et al. 2022]

Chain-of-thought (COT) Prompting

Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ✗

Chain-of-Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9. ✓

[Wei et al. 2022]

Chain-of-thought (COT) Prompting

Step-by-step

- COT requires ICL examples explicitly enumerating the reasoning steps
- Turn out reasoning steps can often be elicited without ICL examples
- Main idea: just “tell” the model to reason in steps

[Kojima et al. 2022]

Chain-of-thought (COT) Prompting

Step-by-step

(a) Few-shot

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A:

(Output) The answer is 8. ✗

(b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are 16 / 2 = 8 golf balls. Half of the golf balls are blue. So there are 8 / 2 = 4 blue golf balls. The answer is 4. ✓

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: The answer (arabic numerals) is

(Output) 8 ✗

(d) Zero-shot-CoT (Ours)

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: ***Let's think step by step.***

(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓

[Kojima et al. 2022]

Chain-of-thought (COT) Prompting

Step-by-step

- COT requires ICL examples explicitly enumerating the reasoning steps
- Turn out reasoning steps can often be elicited without ICL examples
- Main idea: just "tell" the model to reason in steps
- Challenge: the answer is often entangled in the reasoning text — how to extract it?

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: ***Let's think step by step.***

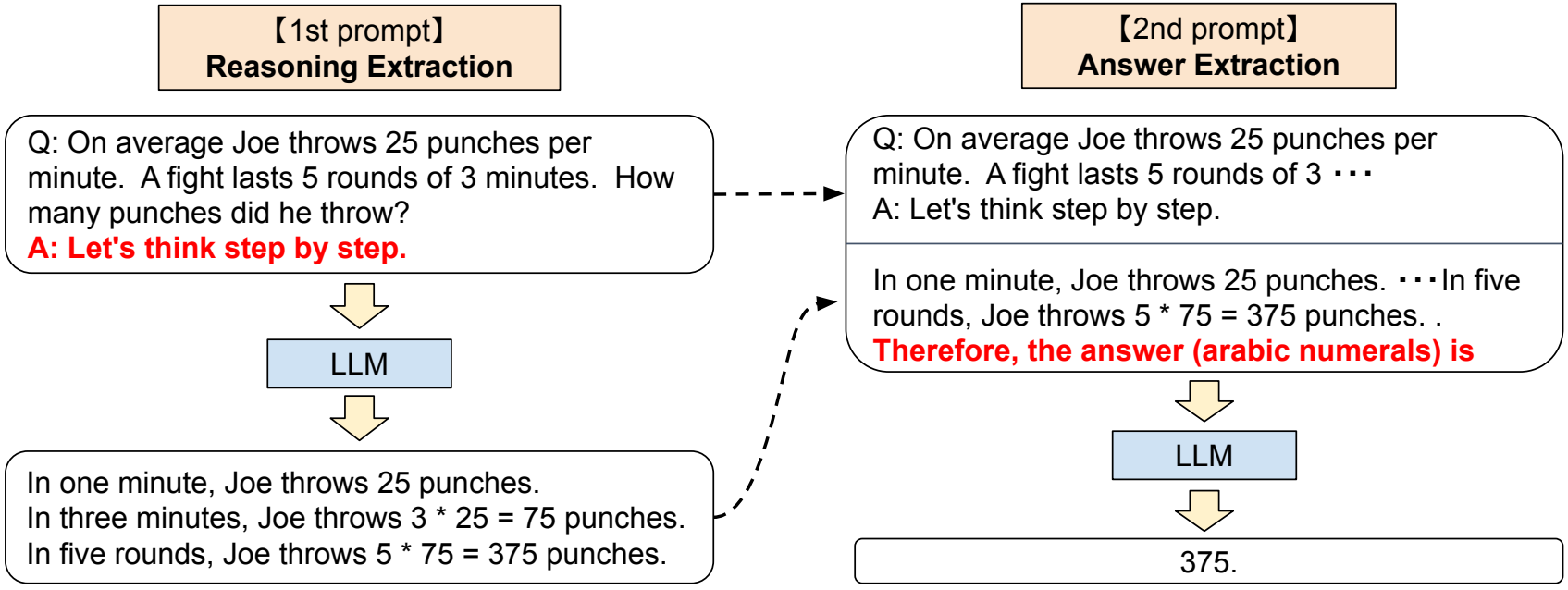
(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓

[Kojima et al. 2022]

Chain-of-thought (COT) Prompting

Step-by-step

- Main idea: just “tell” the model to reason in steps
- Challenge: the answer is often entangled in the reasoning text — how to extract it? → just use an LLM 😀

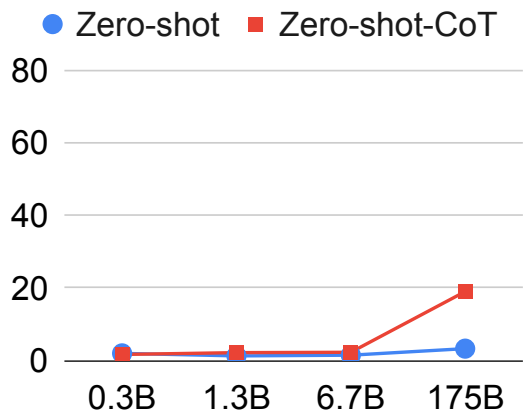


[Kojima et al. 2022]

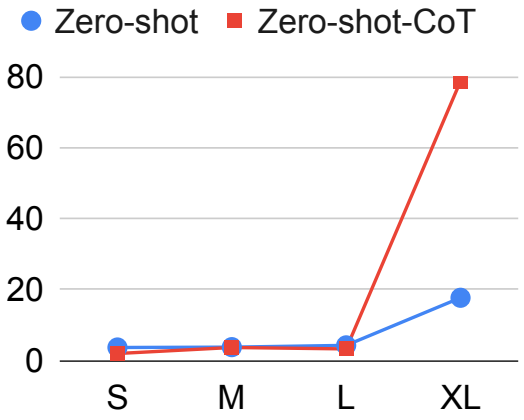
Chain-of-thought (COT) Prompting

Step-by-step

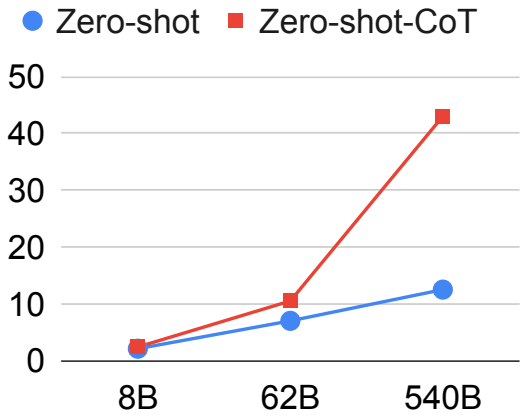
- Main idea: just “tell” the model to reason in steps
- Can significantly outperform zero-shot prompting with very large models
- But requires no ICL examples



(a) MultiArith on Original GPT-3



(b) MultiArith on Instruct GPT-3



(c) GMS8K on PaLM

[Kojima et al. 2022]

Chain-of-thought (COT) Prompting

Step-by-step

- There is no one magical prompt
- Empirically, the is a set of instructive prompts that are roughly equivalent

No.	Category	Template	Accuracy
1	instructive	Let's think step by step.	**78.7**
2		First, (*1)	77.3
3		Let's think about this logically.	74.5
4		Let's solve this problem by splitting it into steps. (*2)	72.2
5		Let's be realistic and think step by step.	70.8
6		Let's think like a detective step by step.	70.3
7		Let's think	57.5
8		Before we dive into the answer,	55.7
9		The answer is after the proof.	45.7
10	misleading	Don't think. Just feel.	18.8
11		Let's think step by step but reach an incorrect answer.	18.7
12		Let's count the number of "a" in the question.	16.7
13		By using the fact that the earth is round,	9.3
14	irrelevant	By the way, I found a good restaurant nearby.	17.5
15		Abrakadabra!	15.5
16		It's a beautiful day.	13.1
-		(Zero-shot)	17.7

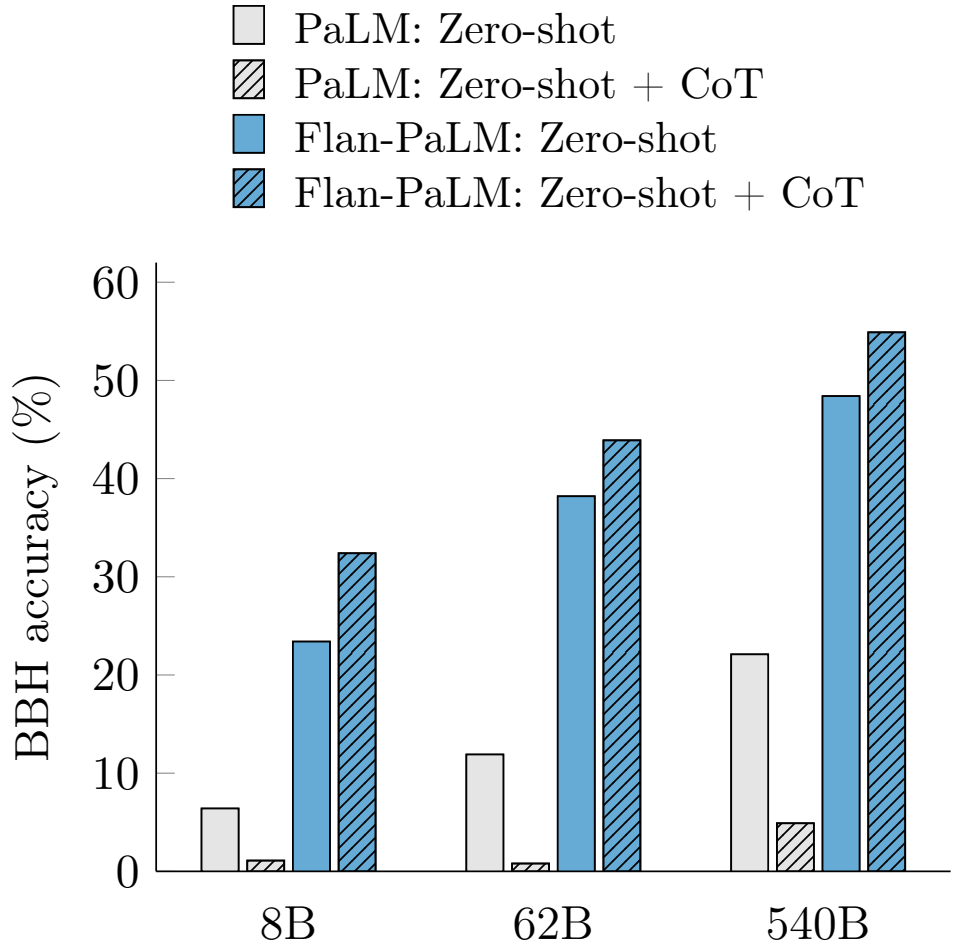
Table 5: Robustness study of Few-shot-CoT against examples. When the examples are from entirely different tasks, the performance generally becomes worse, but when the answer formats are matched (i.e. CommonsenseQA to AQUA-RAT, multiple-choice), the performance loss is less severe. †CommonsenseQA samples are used in this variation

[Kojima et al. 2022]

Chain-of-thought (COT) Prompting

Fine-tuning

- COT can also be used for fine-tuning
- And can increase zero-shot step-by-step performance



[Chung et al. 2022]

Acknowledgements

- Prompting slides are inspired by slides from UT Austin CS 388 by Greg Durrett

- COT slides are inspired in addition by slides from Berkeley CS 288 by Alane Suhr and Dan Klein