Annotated Data

Working with LLMs: Prompting
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 changes to its 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 $\bar{x}$ in a verbalizer template $\nu$
- The template controls the output

$\bar{x} = \text{the movie’s acting could’ve been better, but the visuals and directing were top-notch.}$

$\nu(\bar{x}) = \text{Review: the movie’s acting could’ve been better, but the visuals and directing were top-notch. Out of positive, negative, or neutral this review is}$

LLM

neutral $\bar{y}$
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 $\bar{x}$ in a verbalizer template $\nu$
- The template controls the output

$\bar{x} =$ the movie’s acting could’ve been better, but the visuals and directing were top-notch.

$\nu(\bar{x}) =$ Review: the movie’s acting could’ve been better, but the visuals and directing were top-notch. Out of positive, negative, or neutral this review is

LLM

3 stars $\bar{y}$
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
    \[
    \arg \max_{\bar{y} \in \{1,2,3,4\}} p(\bar{y} | v(\bar{x}))
    \]
  - If the label is a single token (|\bar{y}| = 1), just compare next token probabilities over labels
  - Otherwise, compute \( p(\bar{y} | v(\bar{x})) \) 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

\[
\arg \max_{\bar{y} \in \{1, 2, 3, 4\}} p(\bar{y} | v(\bar{x}))
\]

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

  - Otherwise, compute \( p(\bar{y} | v(\bar{x})) \) 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: it increases alternative 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.
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.
Zero-shot Prompting
Sensitivity and Variability

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

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

([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).
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

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

Surface Form Competition: Why the Highest Probability Answer Isn’t Always Right

Ari Holtzman
Peter West
Vered Shwartz
Yejin Choi
Luke Zettlemoyer

1 Paul G. Allen School of Computer Science & Engineering, University of Washington
2 Allen Institute for Artificial Intelligence

Abstract

Large language models have shown promising results in zero-shot settings (Brown et al., 2020; Radford et al., 2019). For example, they can perform multiple choice tasks simply by conditioning on a question and selecting the answer with the highest probability. However, ranking by string probability can be problematic due to surface form competition—wherein different surface forms compete for probability mass, even if they represent the same underlying concept in a given context, e.g., “computer” and “PC.” Since probability mass is finite, this lowers the probability of the correct answer, due to competition from other strings that are valid answers (but not one of the multiple choice options).

We introduce Domain Conditional Pointwise Mutual Information, an alternative scoring function that directly compensates for surface form competition by simply reweighing each option according to its a priori likelihood within the context of a specific task. It achieves consistent gains in zero-shot performance over both calibrated (Zhao et al., 2021) and uncalibrated scoring functions on all GPT-2 and GPT-3 models on a variety of multiple choice datasets.

Introduction

Despite the impressive results large pretrained language models have achieved in zero-shot settings (Brown et al., 2020; Radford et al., 2019), we argue that current work underestimates the zero-shot capabilities of these models on classification tasks. This is in large part due to surface form competition—a property of generative models that causes probability to be rationed between different valid strings, even ones that differ trivially, e.g., by capitalization alone. Such competition can be largely removed by scoring choices according to Domain Conditional Pointwise Mutual Information (PMI), which reweighs scores by how much more likely a hypothesis (answer) becomes given a premise (question) within the specific task domain.

Specifically, consider the example question (shown in Figure 1): “A human wants to submerge himself in water, what should he use?” with multiple choice options “Coffee cup”, “Whirlpool bath”, “Cup”, and “Puddle.” From the given options, “Whirlpool bath” is the only one that makes sense. Yet, other answers are valid and easier for a language model to generate, e.g., “Bathtub” and “A...”

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).
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 \( \nu \) 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

\( \bar{x} \) = the movie’s acting could’ve been better, but the visuals and directing were top-notch.

Review: The cinematography was stellar; great movie!
Sentiment (positive or negative): positive
Review: The plot was boring and the visuals were subpar.
Sentiment (positive or negative): negative
Review: The movie’s acting could’ve been better, but the visuals and directing were top-notch.
Sentiment (positive or negative):

\( \bar{y} \) = positive
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:
2 cheese => ...........................................
```

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:
2 sea otter => loutre de mer
3 cheese => ...........................................
```

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:
2 sea otter => loutre de mer
3 peppermint => menthe poivrée
4 plush giraffe => girafe peluche
5 cheese => ...........................................
```

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

```
1 sea otter => loutre de mer
2 peppermint => menthe poivrée
3 plush giraffe => girafe peluche
```

[Figure from Brown et al. 2020]
In-context Learning (ICL)

Performance

- Providing ICL examples almost always leads to significant improvements.

![Aggregate Performance Across Benchmarks](image)

**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.
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.

[21]

[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.
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.
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).
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.
In-context Learning (ICL)

Analysis

- In some cases, the label correctness actually matter 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.
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
Chain-of-thought (COT) Prompting

**Standard Prompting**

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?

A: The answer is 27. ✗

**Chain-of-Thought Prompting**

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?

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. ✓
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. X

(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 X

(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. ✓
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. ✓
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 😄

Q: On average Joe throws 25 punches per minute. A fight lasts 5 rounds of 3 minutes. How many punches did he throw?

A: Let's think step by step.

In one minute, Joe throws 25 punches.
In three minutes, Joe throws $3 \times 25 = 75$ punches.
In five rounds, Joe throws $5 \times 75 = 375$ punches.

Therefore, the answer (arabic numerals) is 375.
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 not ICL examples

Figure 3: Model scale study with various types of models. S: text-ada-001, M: text-babbage-001, L: text-curie-001, XL: text-davinci-002. See Appendix A.3 and E for the detail.

Table 3: Examples generated by Zero-Shot-CoT on CommonsenseQA for Error Analysis.

(a) MultiArith on Original GPT-3
(b) MultiArith on Instruct GPT-3
(c) GMS8K on PaLM

Comparison with other baselines
Table 2 compares the performances on two arithmetic reasoning benchmarks (MultiArith and GSM8K) across Zero-shot-CoT and baselines. The large gap between standard prompting (1st block) and chain of thought prompting (2nd block) suggests that these tasks are difficult without eliciting multi-step reasoning. Major improvements are confirmed on both Instruct GPT-3 (text-davinci-002) and PaLM (540B) models (4th block). While Zero-shot-CoT naturally underperforms Few-shot-CoT, it substantially outperforms standard Few-shot prompting with even 8 examples per task. For GSM8K, Zero-shot-CoT with Instruct GPT-3 (text-davinci-002) also outperforms finetuned GPT-3 and standard few-shot prompting with large models (PaLM, 540B), reported in Wei et al. [2022] (3rd and 4th block). See App. D for more experiment results with PaLM.

Does model size matter for zero-shot reasoning?
Figure 3 compares performance of various language models on MultiArith / GSM8K. Without chain of thought reasoning, the performance does not increase or increases slowly as the model scale is increased, i.e., the curve is mostly flat. In contrast, the performance drastically increases with chain of thought reasoning, as the model size gets bigger, for Original/Instruct GPT-3 and PaLM. When the model size is smaller, chain of thought reasoning is not effective. This result aligns with the few-shot experiment results in Wei et al. [2022]. Appendix E shows extensive experiment results using wider variety of language models, including GPT-2, GPT-Neo, GPT-J, T0, and OPT. We also manually investigated the quality of generated chain of thought, and large-scale models clearly demonstrate better reasoning (See Appendix B for the sampled outputs for each model).

Error Analysis
To better understand the behavior of Zero-shot-CoT, we manually investigated randomly selected examples generated by Instruct-GPT3 with Zero-shot-CoT prompting. See Appendix C for examples, where some of the observations include: (1) In commonsense reasoning (CommonsenseQA), Zero-shot-CoT often produces flexible and reasonable chain of thought even when the final prediction is not correct. Zero-shot-CoT often output multiple answer choices when the model find it is difficult to narrow it down to one (see Table 3 for examples). (2) In arithmetic reasoning, Zero-shot-CoT often produces flexible and reasonable chain of thought even when the final prediction is not correct. Zero-shot-CoT often output multiple answer choices when the model find it is difficult to narrow it down to one (see Table 3 for examples).
Chain-of-thought (COT) Prompting
Step-by-step

• There is no one magical prompt

• Empirically, there is a set of instructive prompts that are roughly equivalent

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

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.
Chain-of-thought (COT) Prompting

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

![Bar chart showing BBH accuracy for different models and configurations](image-url)
Acknowledgements

- Prompting slides are inspired by Greg Durrett’s CS 388 slides at UT Austin
- COT slides are inspired in addition by Alane Suhr’s NLP class at Berkeley