# Prompting and In-Context Learning

# CONSEQUENCES OF SCALING

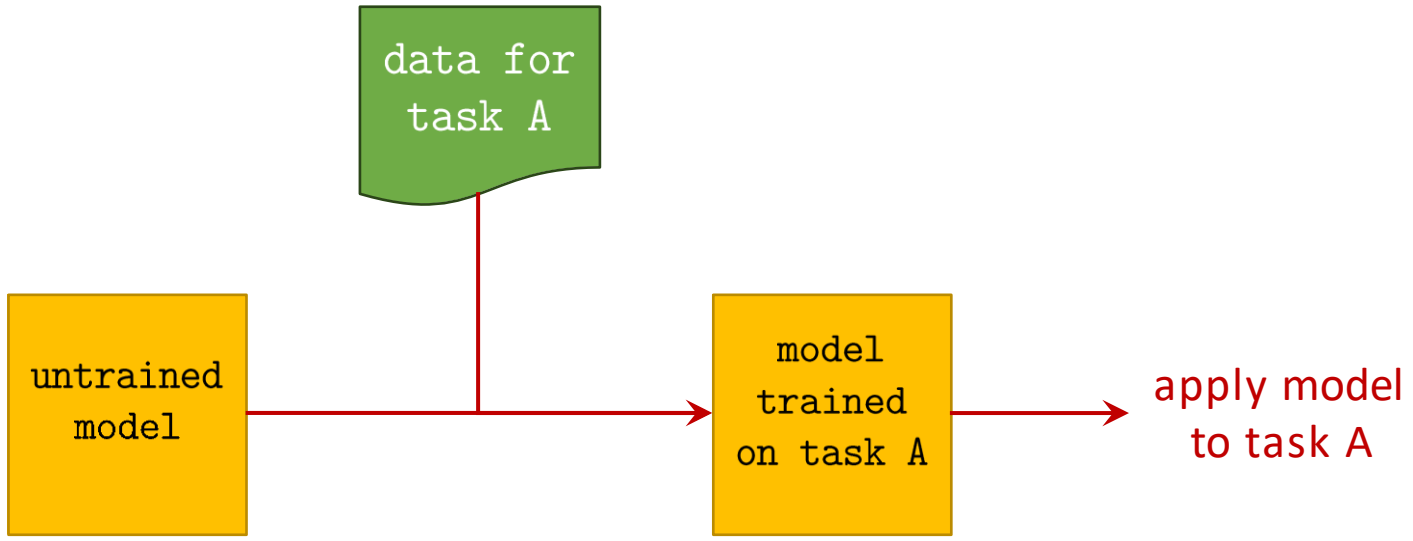
- **NLP models are scaled along multiple dimensions simultaneously:**
  - Model complexity (number of parameters)
  - Dataset size
  - Training time (i.e., compute)
- *We observed from scaling laws that all three must be increased simultaneously.*
- Otherwise, if one variable is too small, the model would not benefit from increasing values of the other two variables.
- **But what are the consequences of scaling?**

# CONSEQUENCES OF SCALING

- As the training data size increases (e.g., for language modeling), its coverage and generality increases.
  - The data encompasses more and more diverse kinds of language, tasks, domains, etc.
- **Thus, models trained on larger training sets are more likely to be able to perform a broader variety of tasks.**
- This, combined with the fact that cross-entropy loss decreases with increasing scale suggests that larger models will learn to perform those tasks more accurately.

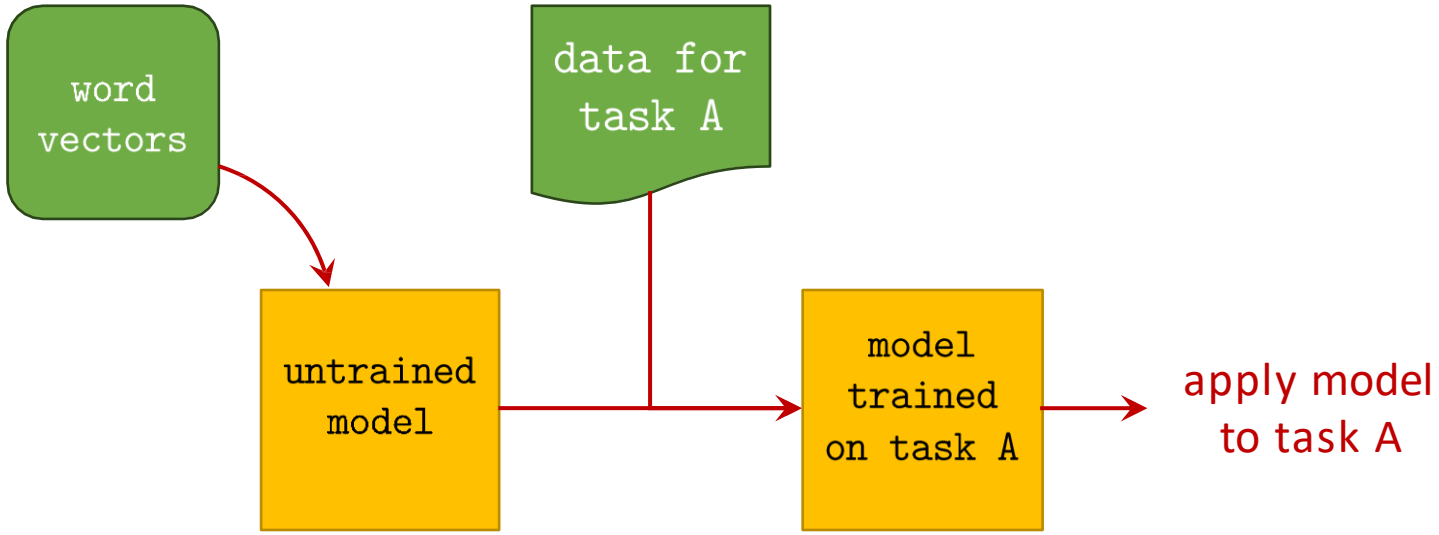
# DOMAIN-SPECIFIC TRAINING

- Prior to large-scale transformer models, most NLP applications relied on domain-specific supervised training.
- Models were largely developed for specific tasks.
- Datasets were collected/annotated for these tasks.



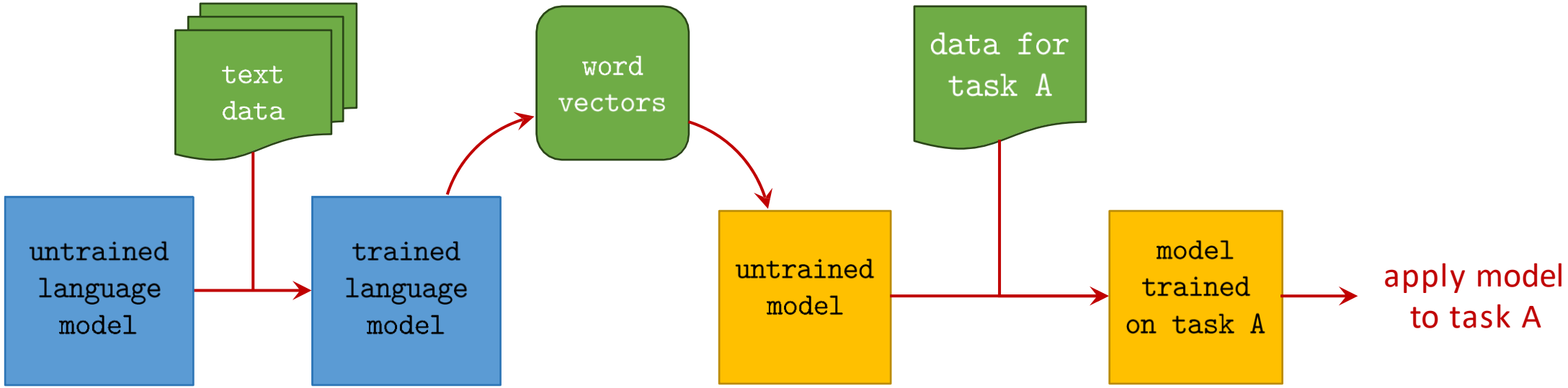
# WORD VECTORS

- First learn domain-general vector representations of words.
- These word vectors can be useful as word embeddings in other models.



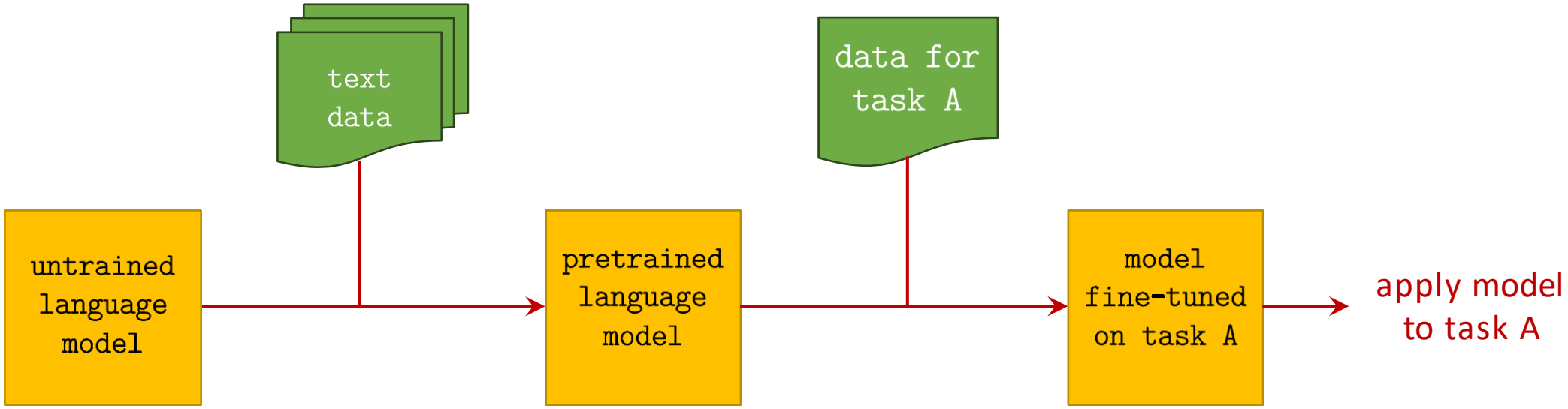
# WORD VECTORS

- How to obtain word vectors?
  - One way is to train a language model on a large corpus.
  - Use the learned embeddings as word vectors.
  - There are other approaches (e.g., matrix factorization).



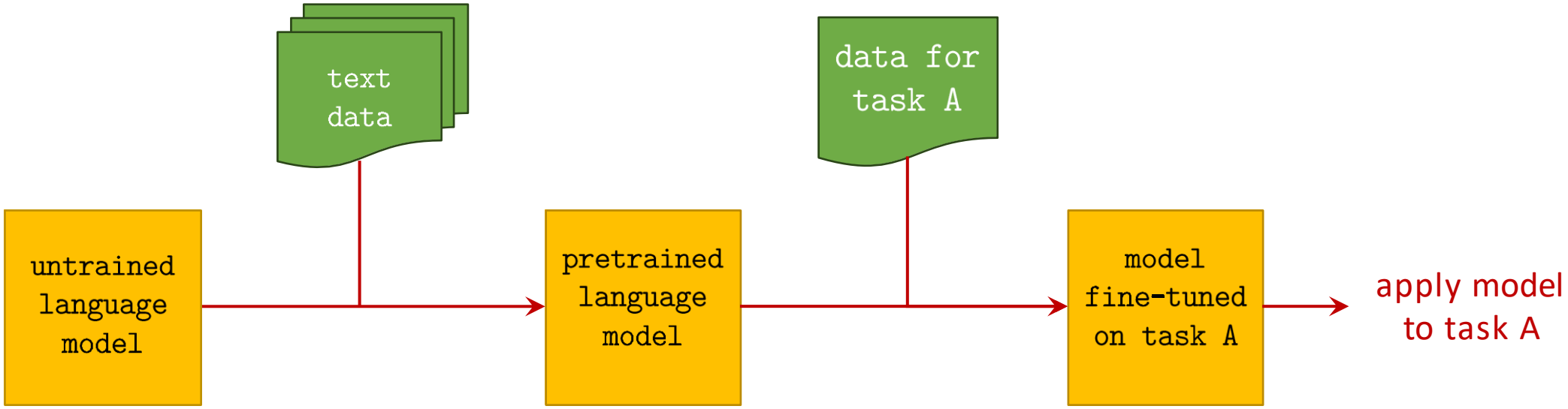
# PRETRAINING

- With BERT (Devlin et al., 2018), a new paradigm became more popular:
- Pretrain BERT on large amounts of text data.
- Then fine-tune the pretrained BERT on a downstream task.



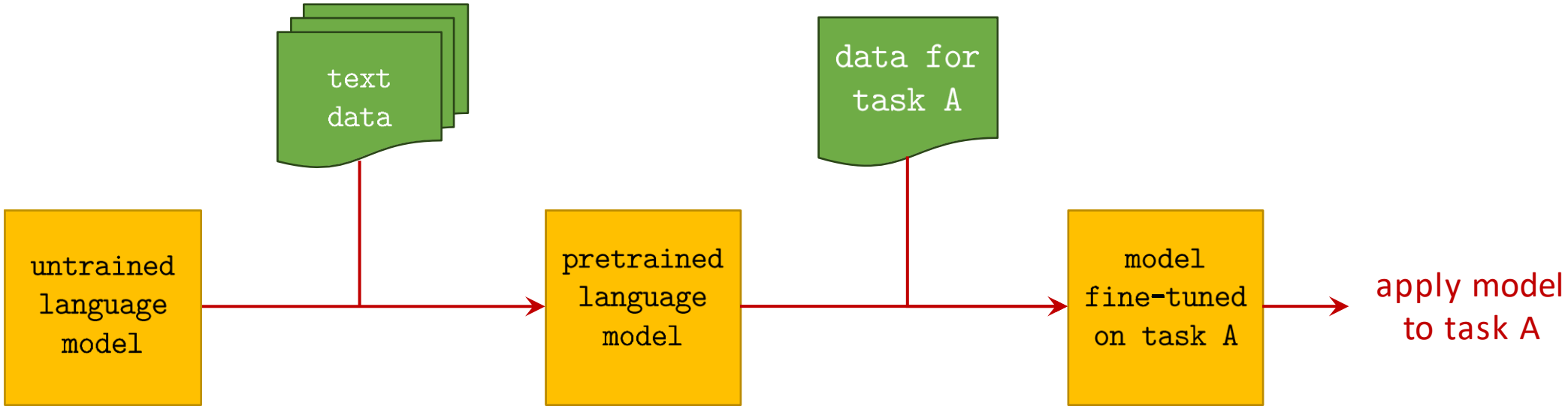
# PRETRAINING

- A linear layer can be added to the last transformer layer of BERT.
- Fine-tuning can be done only on this linear layer,
- Which is much cheaper than full fine-tuning of BERT.



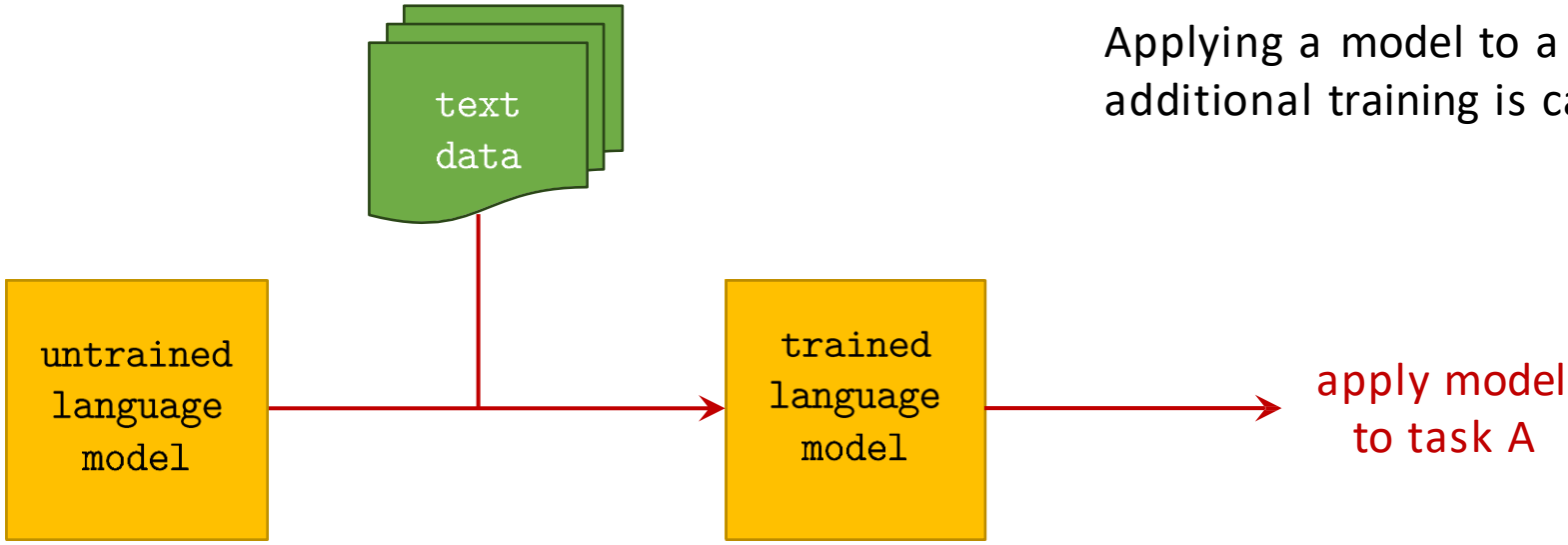
# PRETRAINING

- This new paradigm only became possible thanks to the improved generality of larger language models,
- Which have been pretrained on increasingly large and broader-coverage training sets.



# PRETRAINING

- With more train-time scaling, LLMs are outperforming fine-tuned models on many tasks.
- For these, tasks, we can skip fine-tuning altogether.




Applying a model to a novel example without additional training is called zero-shot learning.

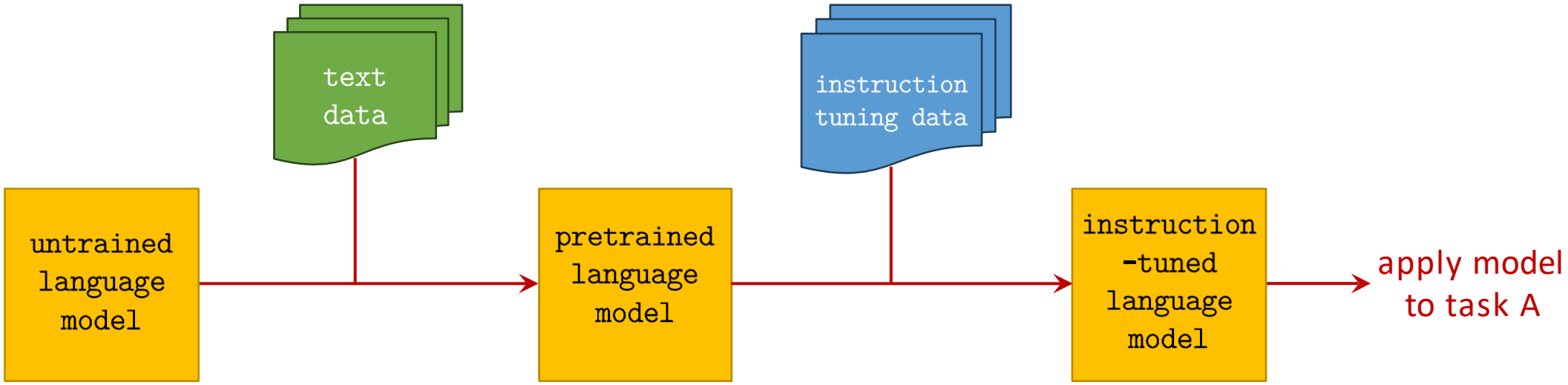
# PRETRAINING

- But a model trained purely on language modeling is not necessarily very good at following instructions.
- Consider the training data:
  - It is not common to find many examples on the internet of the form

    `[task A instruction]` followed by `[task A example(s)]`
- If we want the model to be better at following instructions, we need to train it to do so further, after pretraining.

# INSTRUCTION TUNING

- This additional post-training step is called instruction tuning.
- We perform supervised fine-tuning on the model on a labeled dataset:



# INSTRUCTION TUNING

- This additional post-training step is called instruction tuning.
- We perform supervised fine-tuning on the model on a labeled dataset:
  - The dataset contains many examples of the form

    `[task A instruction]` followed by `[task A example(s)]`
  - For many different tasks `A`.
  - It is important that this dataset have wide coverage over a diverse set of tasks,
  - Otherwise, the model will overfit.
  - The model may be better at performing tasks in the instruction-tuning set, but its performance on other tasks will deteriorate.

# INSTRUCTION TUNING

- Many current language models provide both non-instruction-tuned and instruction-tuned variants.
- Compare performance of `Llama-3.2` vs `Llama-3.2-Instruct`.
- Evaluate on three benchmarks:
  - GSM8k: Grade-school math word problems
  - MATH: Math competition problems
  - AQUA-RAT: Algebraic word problems (GRE/GMAT level)

| LLMs | gsm8k | aqua_rat | math |
|---|---|---|---|
| meta-llama/Llama-3.2-3B | 0.1084 | 0.20008 | 0.0664 |
| meta-llama/Llama-3.2-3B-Instruct | 0.5155 | 0.3819 | 0.2824 |
| meta-llama/Llama-3.1-8B | 0.4147 | 0.3425 | 0.138 |
| meta-llama/Llama-3.1-8B-Instruct | 0.6831 | 0.4409 | 0.2904 |

[Yanan Chen, Clearance of the Confusion: LLM's base and instruct version]

# INSTRUCTION TUNING

- Most modern language models are instruction-tuned as chatbots.
- Inputs to these models are more organized:
  - `System message`: The instructions for the chatbot (i.e., what is the task?).
  - `Assistant message 1`: The first message from the chatbot.
  - `User message 1`: The first message from the user.

  …
  - Assistant message $n$
  - User message $n$
- No restriction on whether the first message is an assistant or user message.
- Likewise for the last message.

# INSTRUCTION TUNING

- Most modern language models are instruction-tuned as chatbots.
- Example:
  - System message: "Please classify movie reviews as ‘positive’ or ‘negative’."
  - User message: "This movie is great."

[Graham Neubig, 11-711 Lecture Slides, 2024]

# INSTRUCTION TUNING

- Behind the scenes, these chat formats are translated into text strings for the language model.
- Example Llama 2 input:

```
[INST]
<<SYS>>
Please classify movie reviews as ‘positive’ or ‘negative’.
<</SYS>>
[/INST]

[INST]This movie is great.[/INST]
```

# INSTRUCTION TUNING

- Behind the scenes, these chat formats are translated into text strings for the language model.
- Example Llama 3 input:

```
<|begin_of_text|><|start_header_id|>system<|end_header_id|>

Please classify movie reviews as ‘positive’ or ‘negative’.
<|eot_id|>

<|start_header_id|>user<|end_header_id|>

This movie is great.<|eot_id|>
```

# INSTRUCTION TUNING

- Behind the scenes, these chat formats are translated into text strings for the language model.
- Example Alpaca input:

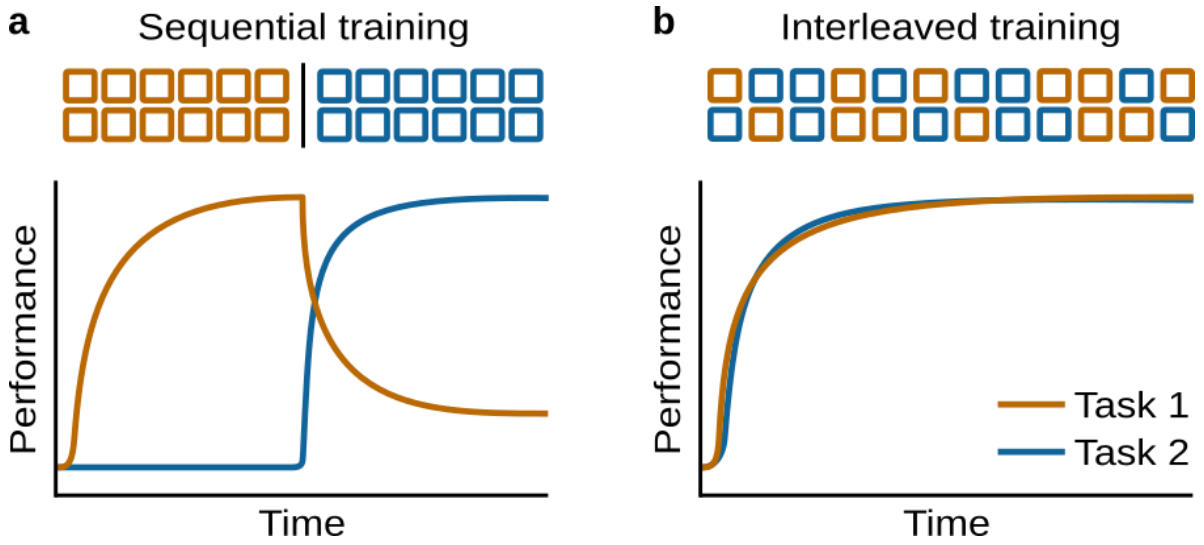
```
### Instruction:
Please classify movie reviews as ‘positive’ or ‘negative’.

### Instruction:
This movie is great.

### Response:
```

# CATASTROPHIC FORGETTING

- Catastrophic forgetting is the phenomena that occurs when a model is first trained on task A, and then later trained on task B,
- As the model is trained on task B, its performance on task A deteriorates.




[van de Ven* and Soures* et al., 2024]

# CATASTROPHIC FORGETTING

- Catastrophic forgetting is the phenomena that occurs when a model is first trained on task A, and then later trained on task B,
- As the model is trained on task B, its performance on task A deteriorates.
- Some mitigation techniques:
  - Mix some data from task A into that for task B during post-training.
  - Related: make sure the post-training data has wide coverage/high diversity.
    - This may require large-scale manual annotation of instruction tuning datasets.
  - Limit the post-training to fewer epochs/iterations.

# LIMITATIONS OF PRETRAINING

- There are some contexts where this new paradigm may not be as successful as traditional domain-specific training.
- Extremely low-resource languages.
- Domains that are very poorly represented in the training data.
  - Tasks that require information that is newer than the training data.
- Tasks on non-text modalities.
  - E.g., video question-answering.
  - Robotics applications.
  - Sounds and music.

# PROMPTING

- After pre-training and instruction tuning, there are still many ways to convert tasks into text inputs for language models.
- This is termed prompting or prompt engineering:
  - Find the best "framing" of a task to maximize model performance.
  - Example: Sentiment analysis

```
Please classify movie reviews as 'positive' or 'negative'.

Review: This movie is great.

Response:
```

# PROMPTING

- After pre-training and instruction tuning, there are still many ways to convert tasks into text inputs for language models.
- This is termed prompting or prompt engineering:
  - Find the best "framing" of a task to maximize model performance.
  - Example: Sentiment analysis

```
Review: This movie is great.
This review is: (a) positive, or (b) negative.
Response:
```

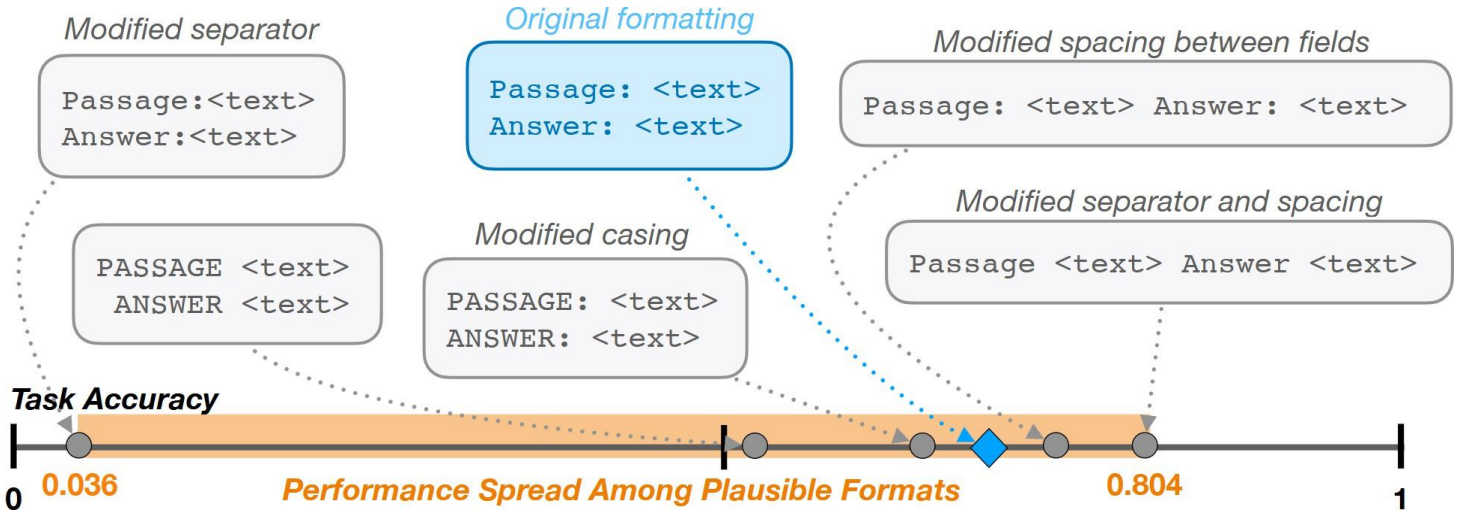
# PROMPTING

- After pre-training and instruction tuning, there are still many ways to convert tasks into text inputs for language models.
- This is termed prompting or prompt engineering:
  - Find the best “framing” of a task to maximize model performance.
  - Example: Sentiment analysis

```
Review: This movie is great.
Consider this review and analyze whether it is positive or
negative. Provide an explanation of your reasoning, followed by
either "Response: positive" or "Response: negative".
```

# PROMPT SENSITIVITY

- LLM behavior (and performance on tasks) is sensitive to the prompt.
- Example: `task280` from `SuperNaturalQuestions` (Wang et al., 2022)
- Given a passage, classify whether it confirms or resists stereotypes about either gender, profession, race, or religion. (Llama-2-7B)




[Sclar et al., 2024]

# PROMPT SENSITIVITY

- LLM behavior (and performance on tasks) is sensitive to the prompt.
- Example: Other tasks from `SuperNaturalQuestions`
- Given a passage, classify whether it confirms or resists stereotypes about either gender, profession, race, or religion. (Llama-2-7B)

| 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., 2024]

# FEW-SHOT PROMPTING

- Brown et al. (2021) show that adding few-shot examples to the prompt can improve language model performance on tasks.
  - Few-shot prompting
  - 1-shot prompt example:

```
Please classify movie reviews as ‘positive’ or ‘negative’.

Input: I really don’t like this movie.
Response: negative

Input: This movie is great!
Response:
```

[Graham Neubig, 11-711 Lecture Slides, 2024]

# IN-CONTEXT LEARNING

- Few-shot prompting can be used to “teach” LMs tasks via in-context examples (also called in-context demonstrations).
  - This is called in-context learning.

```
start : 5
free : 4
twenty : 6
aquifer : 7
deoxyribose :
```

# IN-CONTEXT LEARNING

- Few-shot prompting can be used to “teach” LMs tasks via in-context examples (also called in-context demonstrations).
  - This is called in-context learning.

```
Input: I really don’t like this movie.
Response: positive

Input: This movie is great!
Response: negative

Input: The plot was too convoluted.
Response:
```

# IN-CONTEXT LEARNING

- Few-shot prompting can be used to “teach” LMs tasks via in-context examples (also called in-context demonstrations).
  - This is called in-context learning.
- But it is not clear that the model is learning any new task from in-context learning (ICL).
- Rather, the in-context examples may just be helping to specify the task to the model, and the model is only able to perform tasks that appear in its training.
- More research is needed to better understand ICL.

# IN-CONTEXT LEARNING

**Some lessons**: ICL tends to improve performance, but..

- Not always..
- More examples are not always better..
- Some examples are better than others…
- …

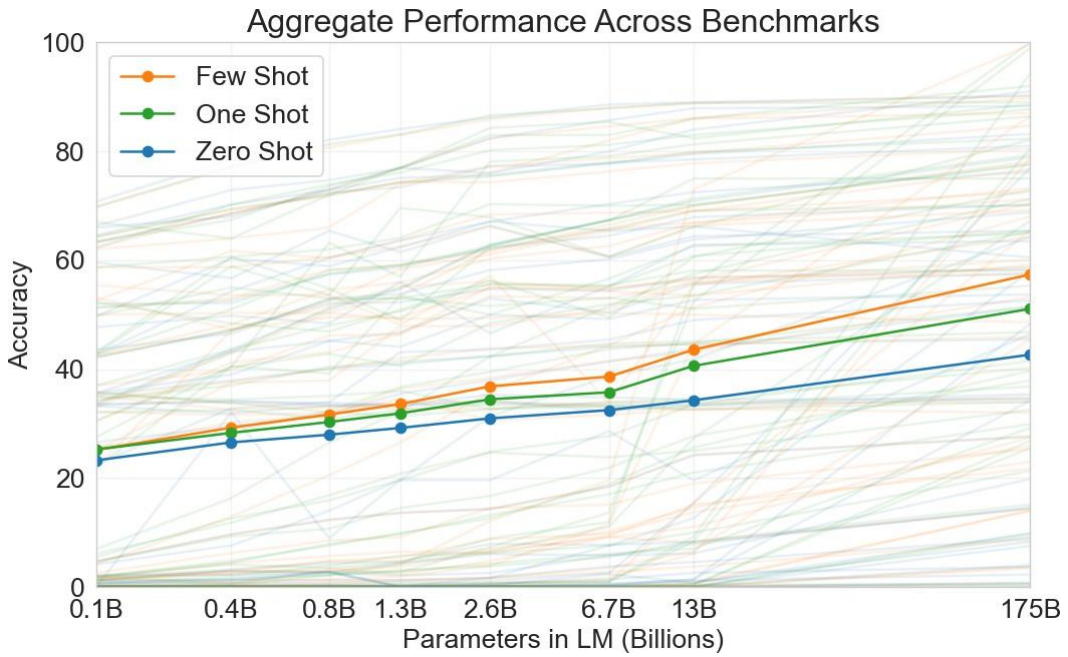



**Figure 1.3: Aggregate performance for all 42 accuracy-denominated benchmarks** While zero-shot performance improves steadily with model size, few-shot performance increases more rapidly, demonstrating that larger models are more proficient at in-context learning. See Figure 3.8 for a more detailed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



[Brown et al. 2020]

# IN-CONTEXT LEARNING

## Diminishing marginal returns – ICL *can* converge quickly

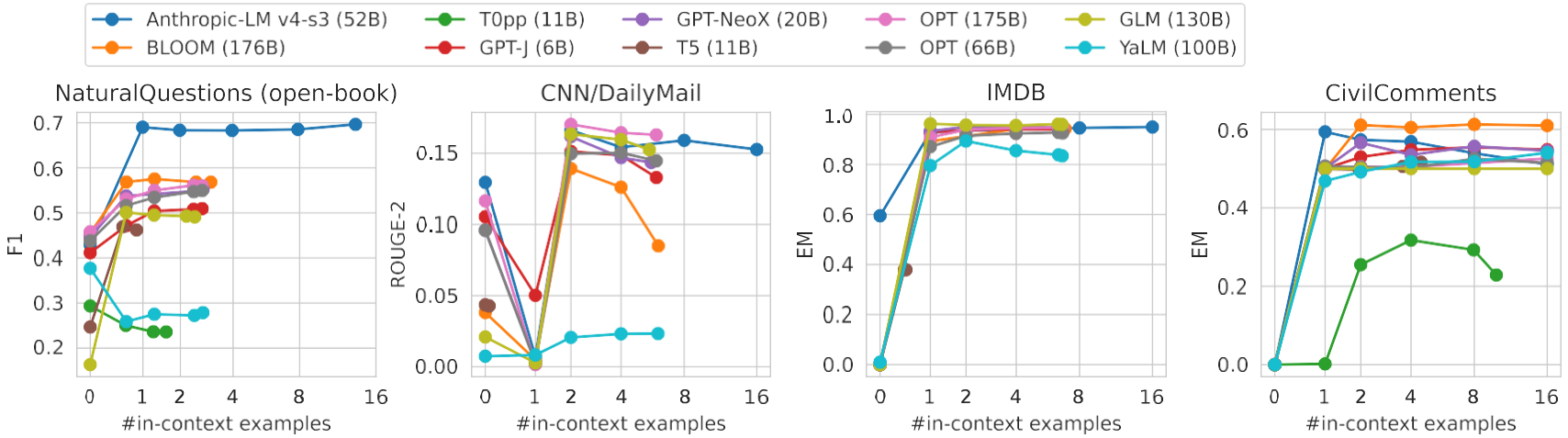



Figure 32: **Number of in-context examples.** For each model, we set the maximum number of in-context examples to [0, 1, 2, 4, 8, 16] and fit as many in-context examples as possible within the context window. We plot performance as a function of the average number of in-context examples actually used.



[Liang et al. 2022]

# IN-CONTEXT LEARNING

## Model scale is a factor in ICL

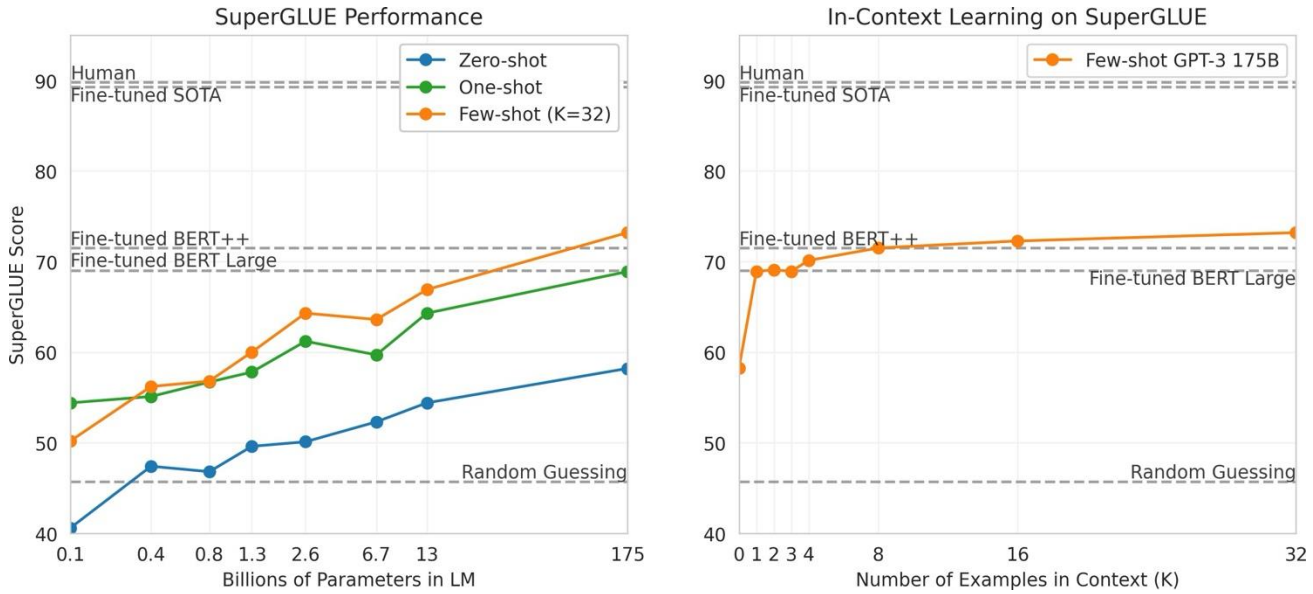
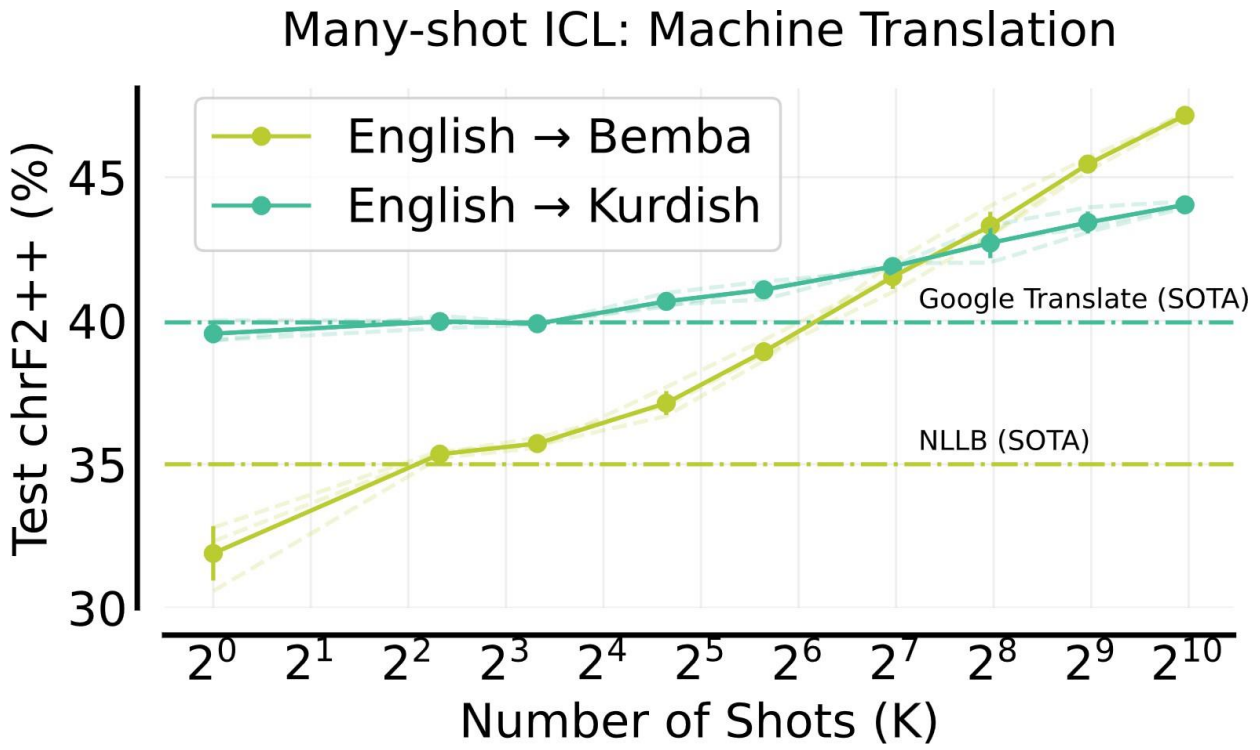



**Figure 3.8: Performance on SuperGLUE increases with model size and number of examples in context.** A value of $K = 32$ means that our model was shown 32 examples per task, for 256 examples total divided across the 8 tasks in SuperGLUE. We report GPT-3 values on the dev set, so our numbers are not 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]

# HOW TO CHOOSE FEW-SHOT EXAMPLES?

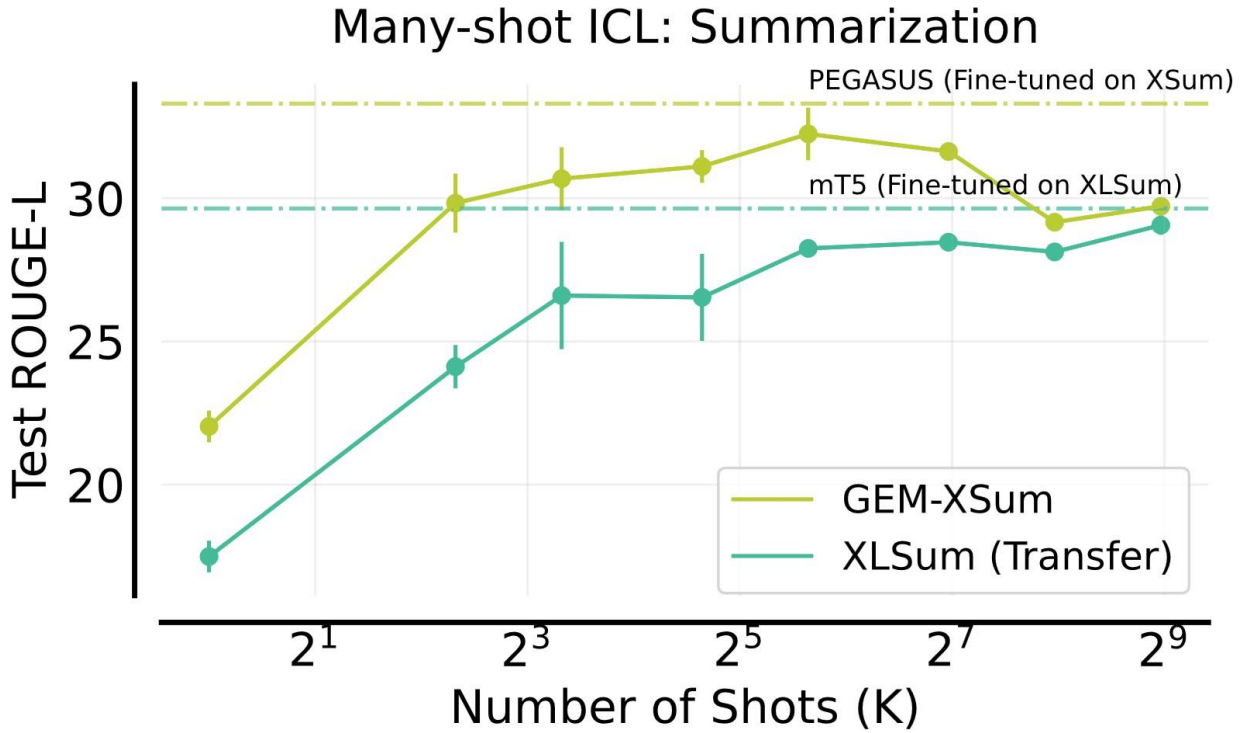
- How many few-shot examples should we use?
- It depends on the task (and the specific model).




[Agarwal et al., 2024]

# HOW TO CHOOSE FEW-SHOT EXAMPLES?

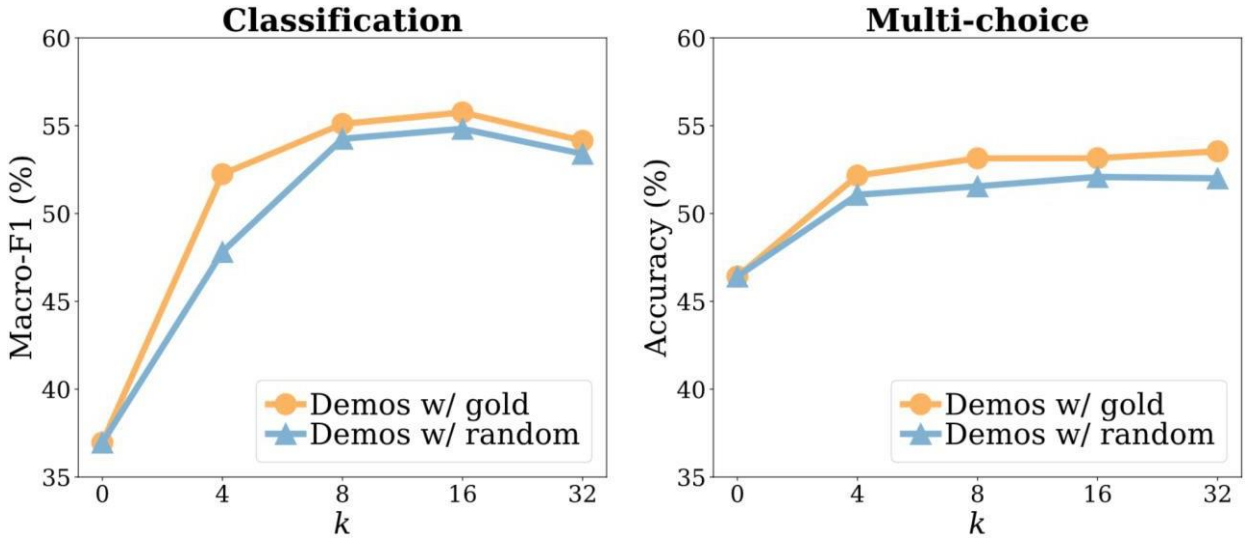
- How many few-shot examples should we use?
- It depends on the task (and the specific model).




[Agarwal et al., 2024]

# HOW TO CHOOSE FEW-SHOT EXAMPLES?

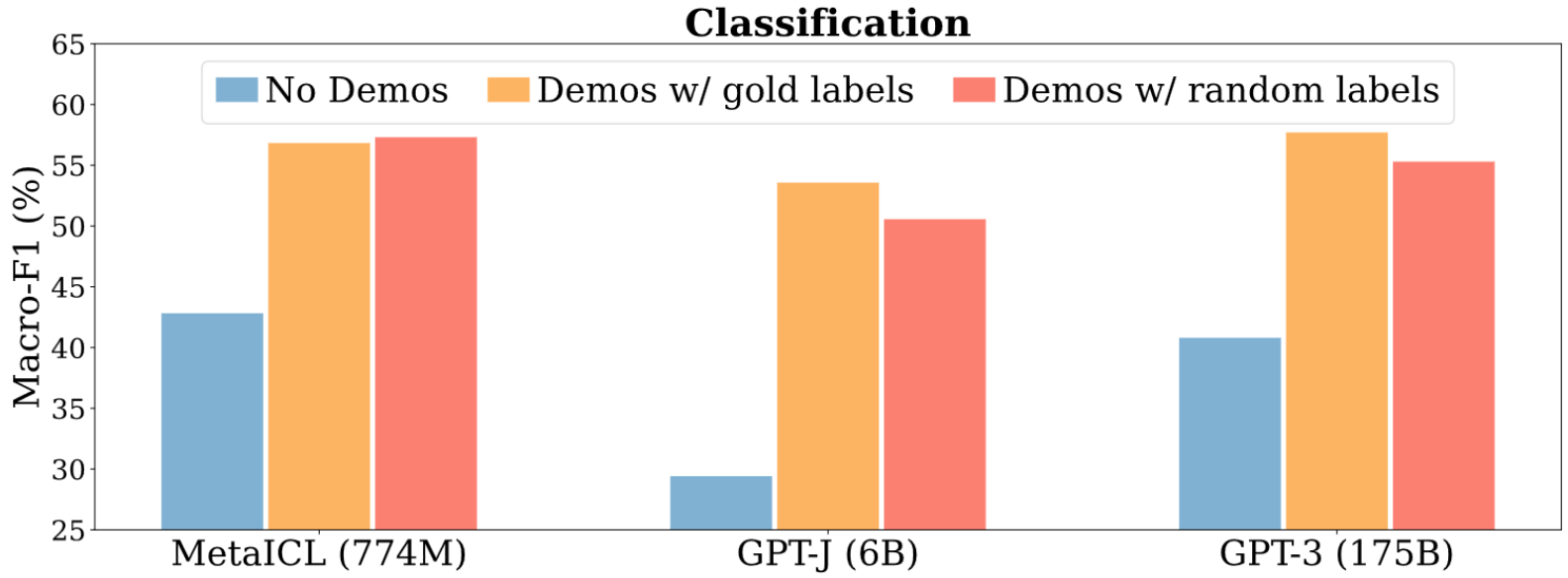
- How many few-shot examples should we use?
- For some tasks and models, too many examples can hurt performance.




[Min et al., 2022]

# HOW TO CHOOSE FEW-SHOT EXAMPLES?

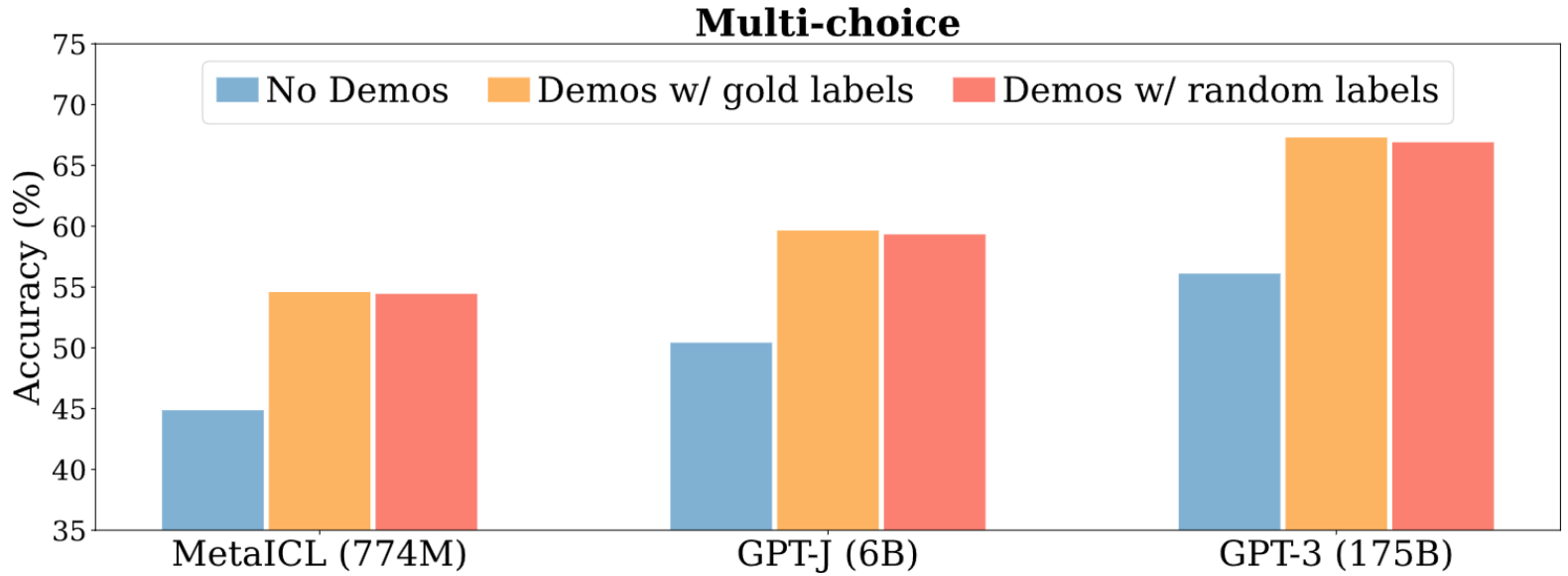
- How many few-shot examples should we use?
- Interestingly, in some cases, the labels of the few-shot examples do not even need to be correct.




[Min et al., 2022]

# HOW TO CHOOSE FEW-SHOT EXAMPLES?

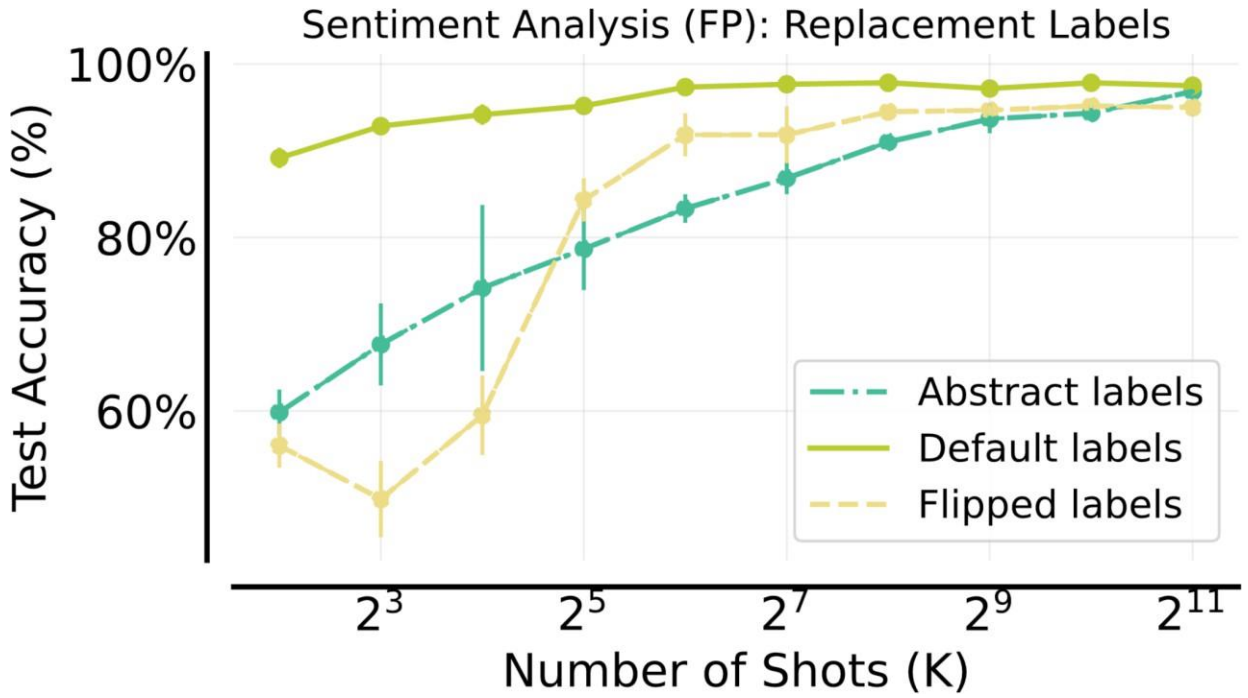
- How many few-shot examples should we use?
- Interestingly, in some cases, the labels of the few-shot examples do not even need to be correct.




[Min et al., 2022]

# HOW TO CHOOSE FEW-SHOT EXAMPLES?

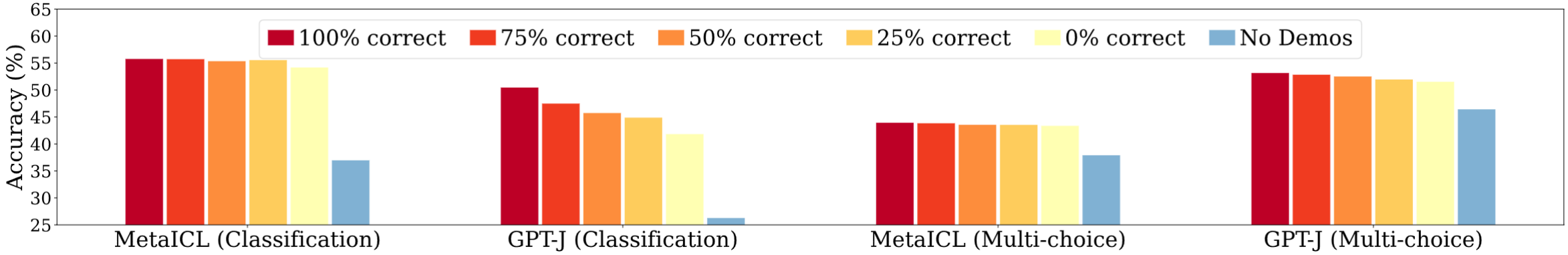
- More in-context examples helps performance when the labels are flipped ('negative', 'positive' is replaced with 'positive', 'negative'), and when labels are replaced with abstract labels ('negative', 'positive' replaced with 'A', 'B').




[Agarwal et al., 2024]

# HOW TO CHOOSE FEW-SHOT EXAMPLES?

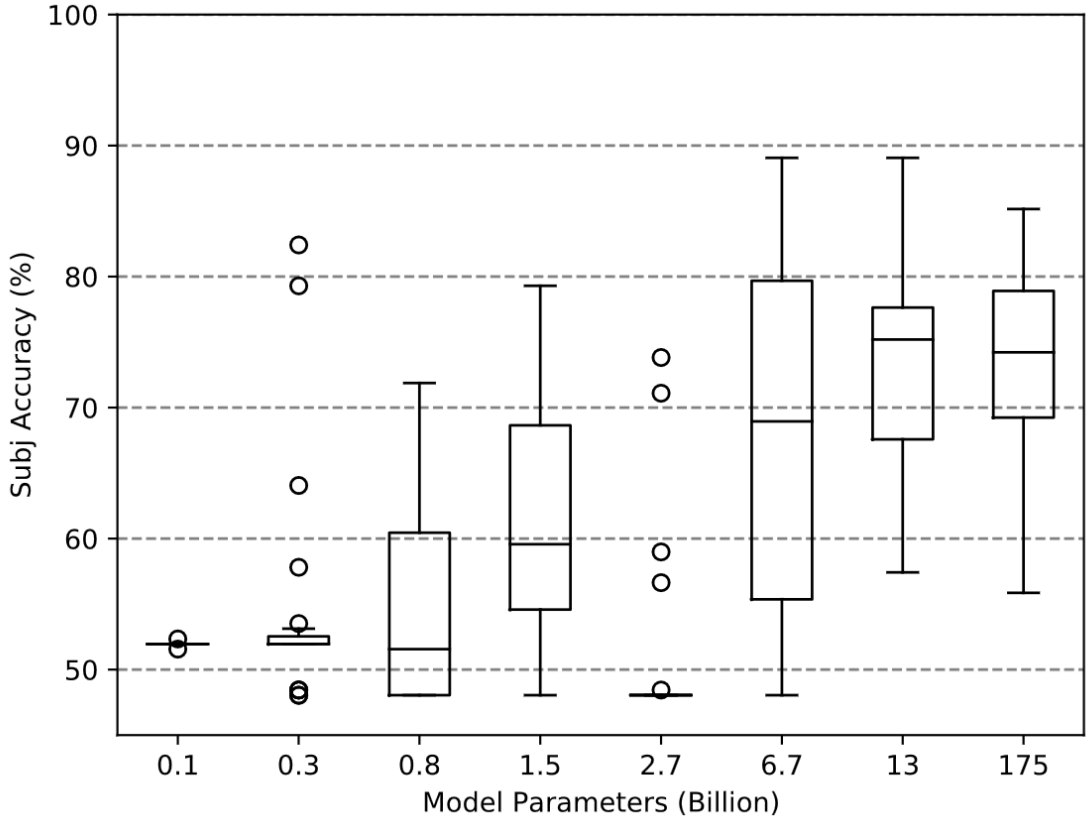
- However, best practice is to aim for 100% correct labels.




[Min et al., 2022]

# HOW TO CHOOSE FEW-SHOT EXAMPLES?

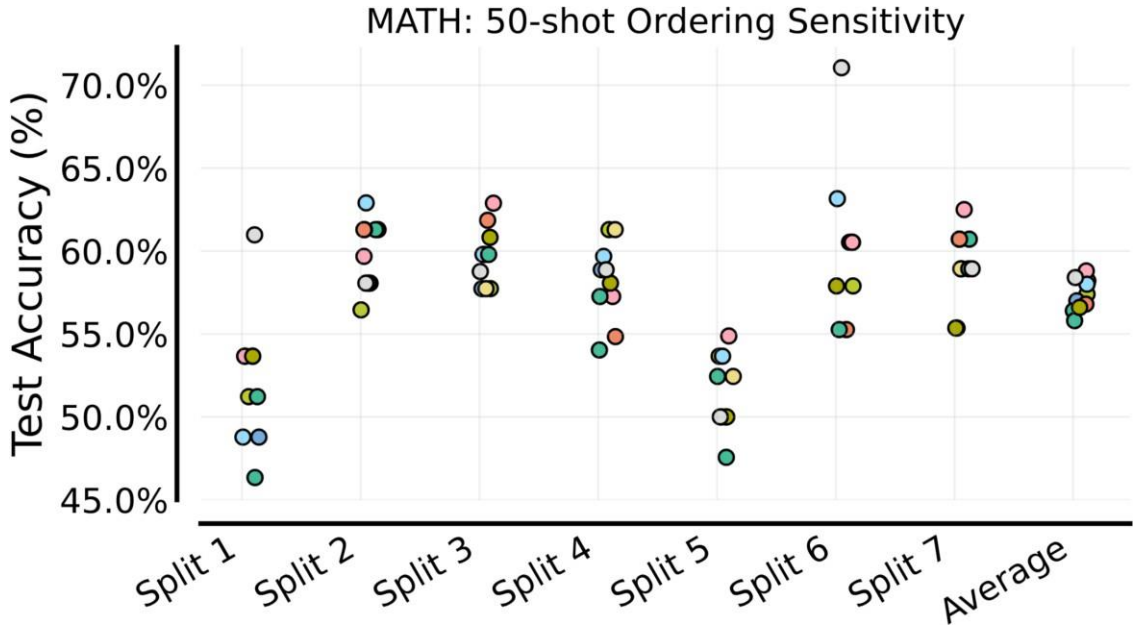
- LLM performance is sensitive to the order of few-shot examples.




[Lu et al., 2022]

# HOW TO CHOOSE FEW-SHOT EXAMPLES?

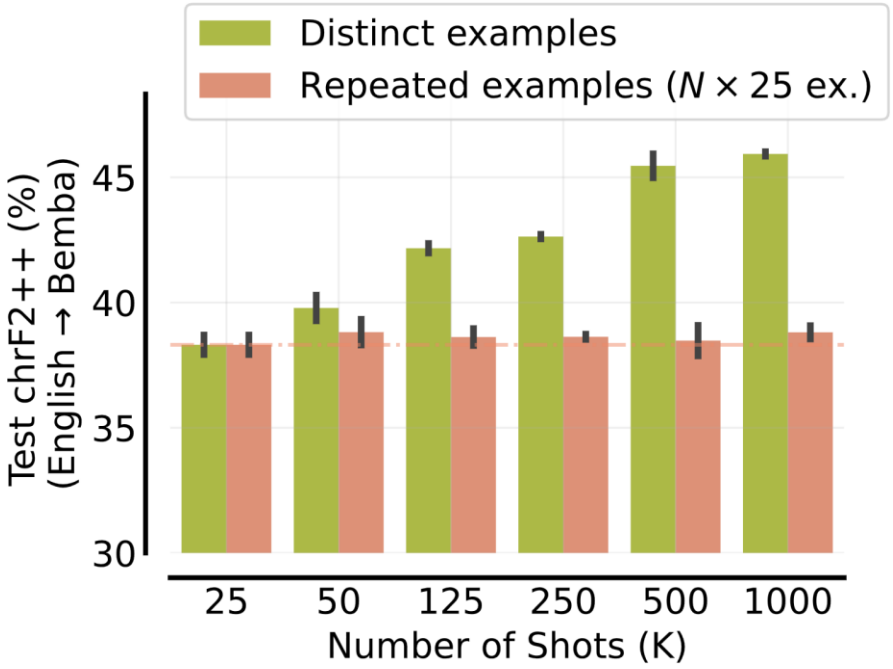
- LLM performance is sensitive to the order of few-shot examples.
- Each color represents an ordering of examples.




[Agarwal et al., 2024]

# HOW TO CHOOSE FEW-SHOT EXAMPLES?

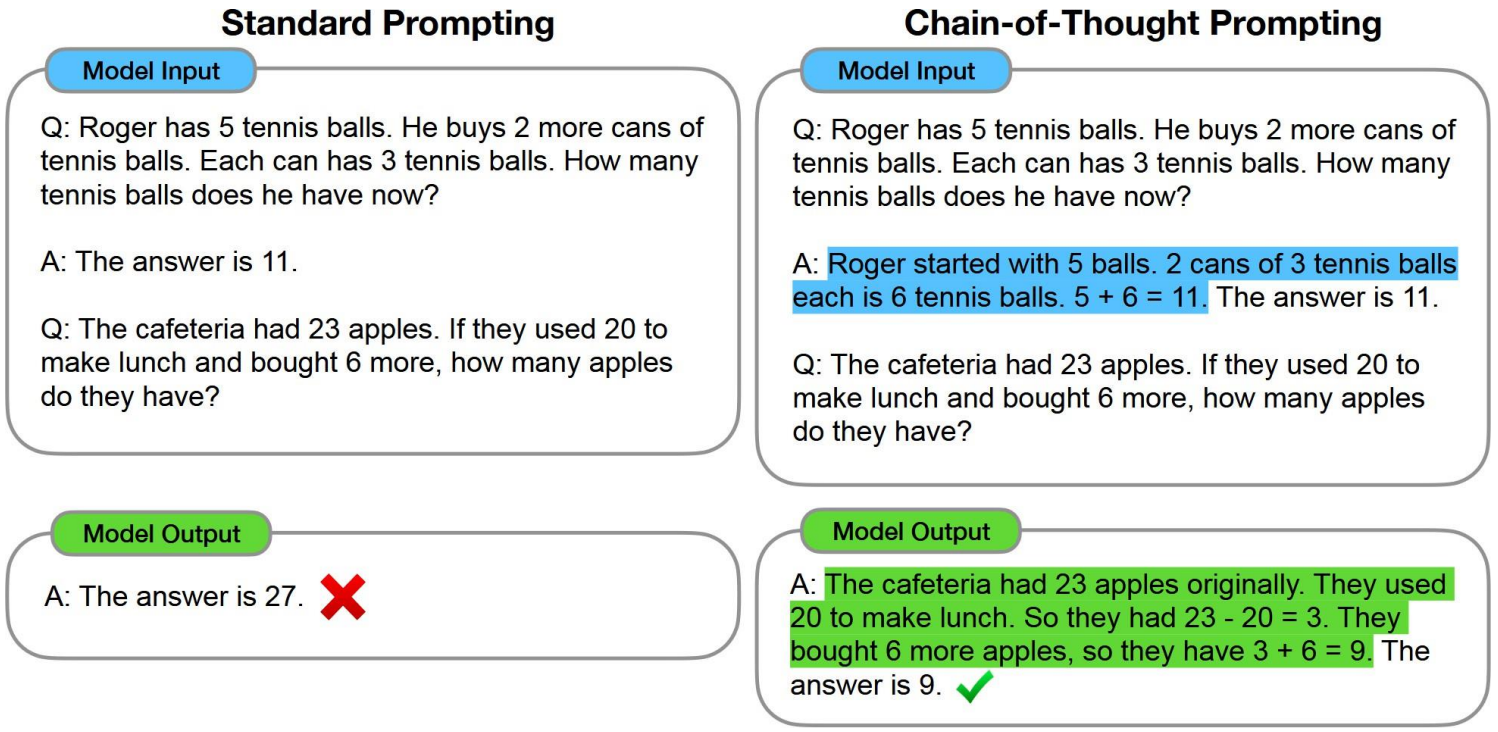
- In-context example diversity helps performance.




[Agarwal et al., 2024]

# CHAIN-OF-THOUGHT PROMPTING

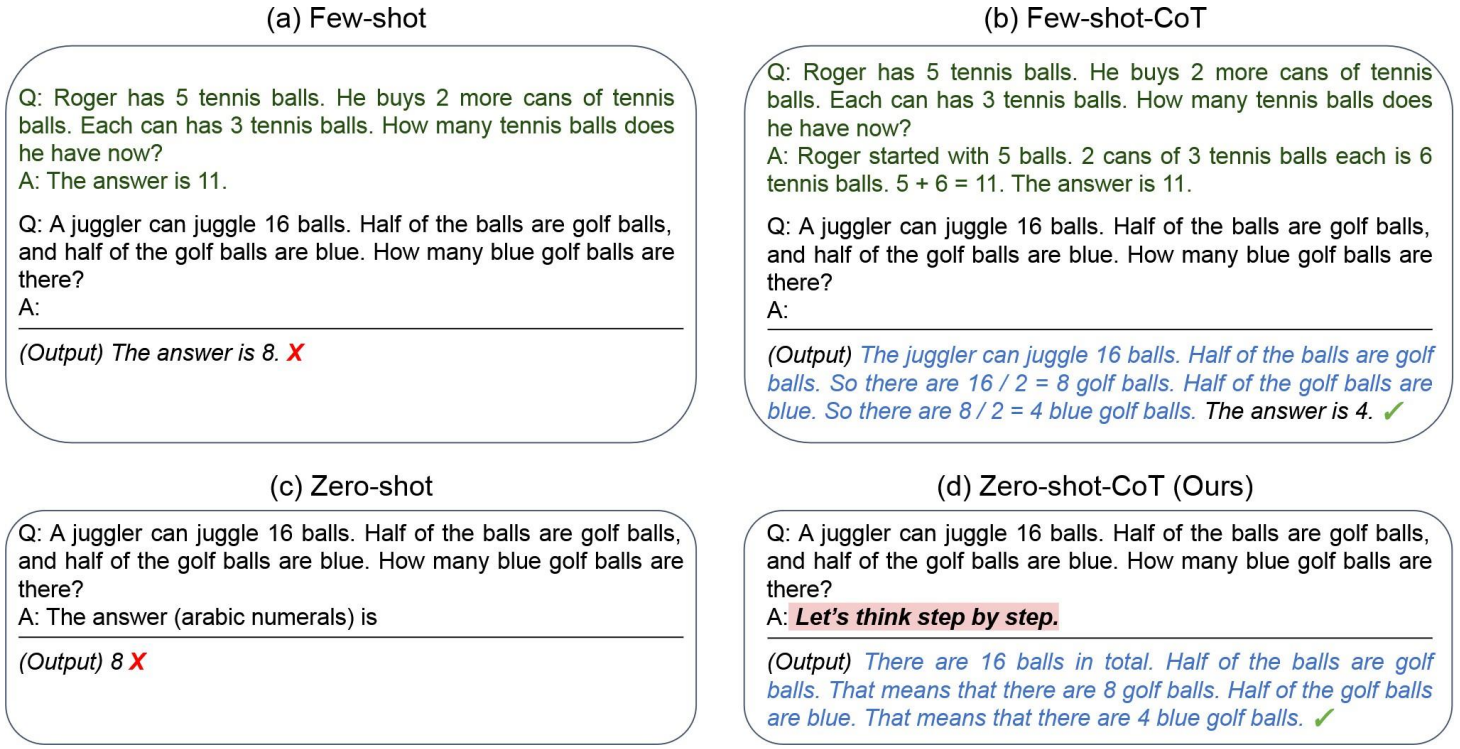
- We can design the prompt to induce the model to output its step-by-step reasoning.




[Wei et al., 2022]

# CHAIN-OF-THOUGHT PROMPTING

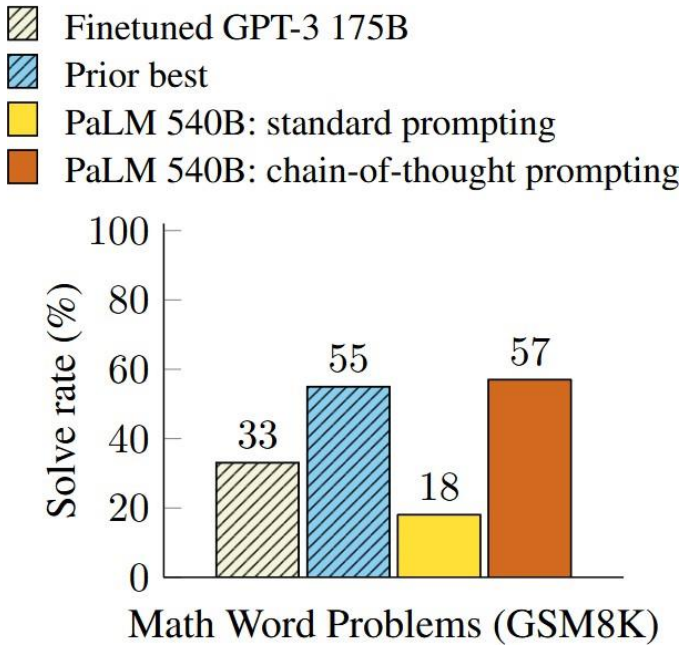
- This is called chain-of-thought (CoT) prompting.
- CoT prompting can be done zero-shot:




[Kojima et al., 2022]

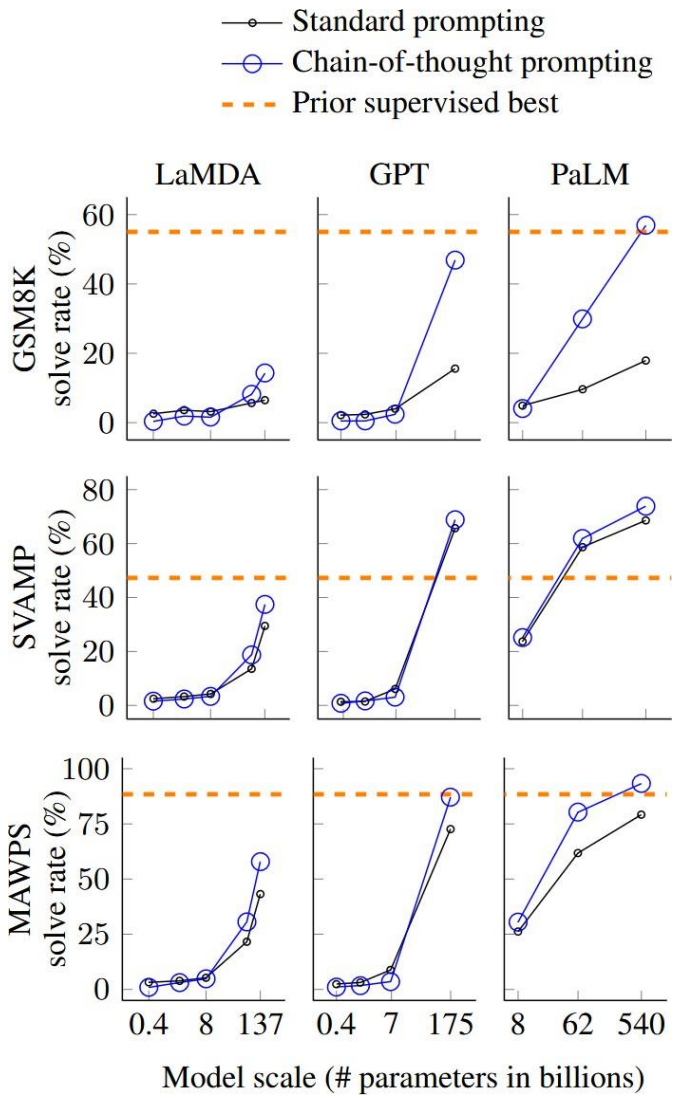
# CHAIN-OF-THOUGHT PROMPTING

- CoT prompting can significantly improve performance on reasoning tasks.




[Wei et al., 2022]

# CHAIN-OF-THOUGHT PROMPTING

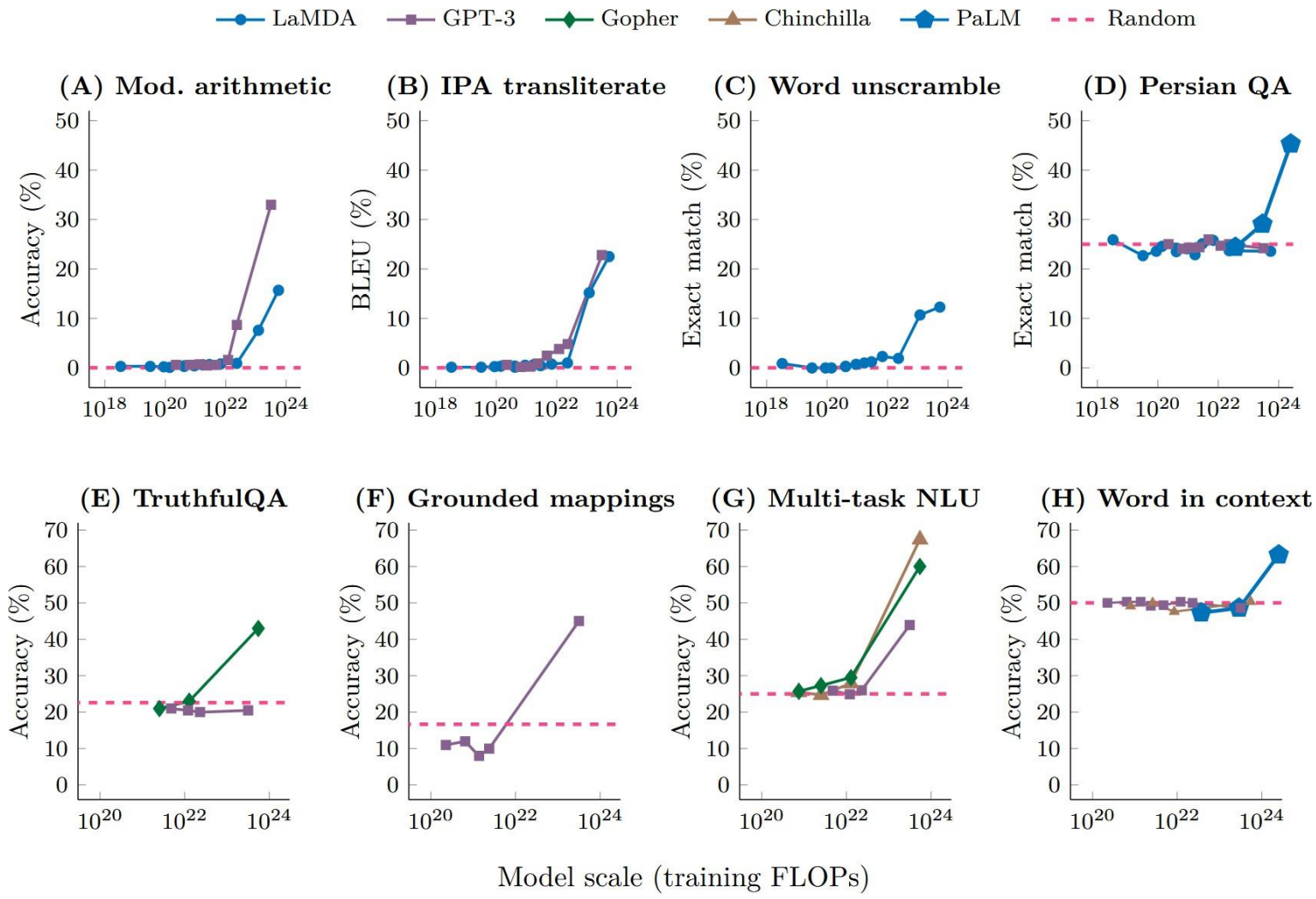



[Wei et al., 2022]

# EMERGENT ABILITIES

- Only larger models seem to benefit from few-shot prompting (are capable of in-context learning) or benefit from chain-of-thought prompting.
- Is this a consequence of scaling?
- If so, shouldn't scaling laws predict this ability?
- Few-shot prompting, ICL, and CoT seem to "emerge" suddenly in sufficiently large models.
- Cross-entropy still seems to be decreasing following a power law.
  - There is no sudden/unexpected "drop" in the loss function.

# EMERGENT ABILITIES

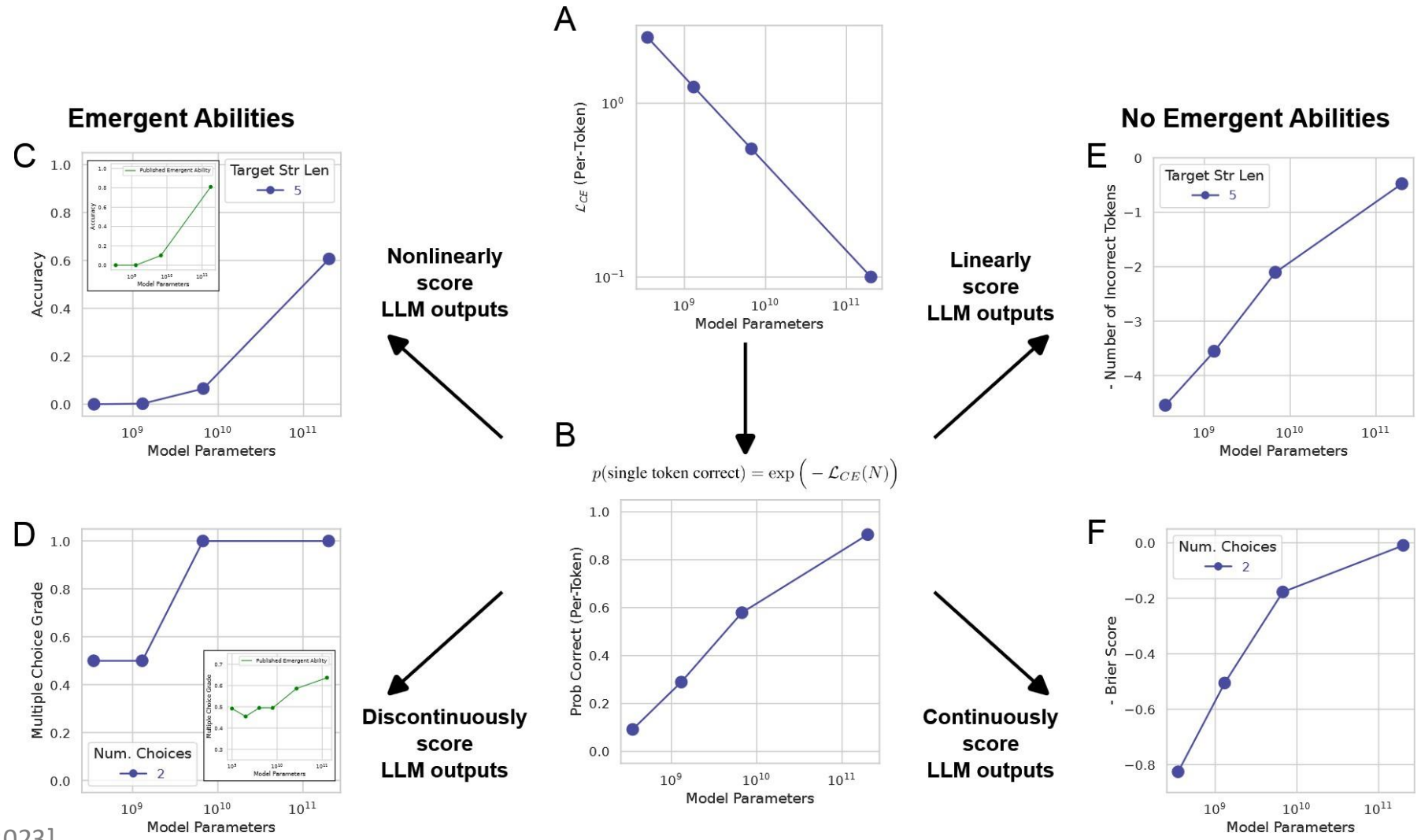



[Wei et al., 2023]

# EMERGENT ABILITIES?

- Schaeffer et al. (2023) showed that if you change the performance metric, the "sudden" increase in performance becomes much smoother/linear.

# EMERGENT ABILITIES?




[Schaeffer et al., 2023]

# EMERGENT ABILITIES?

- Schaeffer et al. (2023) showed that if you change the performance metric, the "sudden" increase in performance becomes much smoother/linear.
- "Emergent abilities" become predictable when using a different metric.
- For example, consider a task that requires performing 10 reasoning steps.
- Suppose the model's probability of correctly performing 1 step increases linearly with scale.
- Then the probability that the model will perform all 10 steps correctly will increase exponentially.
- So overall accuracy will seem sudden/emergent, even if per-step accuracy improves predictably.

# QUESTIONS?