CS388: Natural Language Processing

Lecture 18: Understanding In-Context Learning

Greg Durrett

TEXAS
The University of Texas at Austin

Administrivia

- A5 out today
- Project proposals for independent FPs due Friday
- Midterm grading underway

Context for the rest of the course

- Next few lectures: revisit what we can do with large language models
  - Prompting
  - Factuality of responses
  - Explaining reasoning
  - How do we build ChatGPT? (RLHF)
- After: understand neural nets better
- Finally: miscellaneous modern topics

This Lecture

- Prompting: best practices and why it works
  - Zero-shot prompting: role of the prompt
  - Few-shot prompting (in-context learning): characterizing demonstrations
  - Factuality of responses
- Understanding in-context learning (brief)
  - Induction heads and mechanistic interpretability

Zero-shot Prompting

Zero-shot Prompting

- GPT-3/4/ChatGPT can handle lots of existing tasks based purely on incidental exposure to them in pre-training
  - Example from summarization: the token “tl;dr” (“too long; didn’t read”) is an indicator of summaries in the wild
- We’ll discuss two paradigms: **zero-shot prompting**, where no examples are given to a model (just a text specification), and **few-shot prompting**, where a few examples are given in-context
- Both paradigms can theoretically handle classification, text generation, and more!

Zero-shot Prompting

- Single unlabeled datapoint ***x***, want to predict label *y*

  x = *The movie's acting could've been better, but the visuals and directing were top-notch.*

- Wrap ***x*** in a template we call a verbalizer ***v***

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

GPT-3

neutral

Zero-shot Prompting

- Single unlabeled datapoint ***x***, want to predict label *y*

x = *The movie's acting could've been better, but the visuals and directing were top-notch.*

- Wrap ***x*** in a template we call a verbalizer ***v***

Review: The movie's acting could've been better, but the visuals and directing were top-notch.
On a 1 to 4 star scale, the reviewer would probably give this movie

GPT-3

3 stars.

Ways to do classification

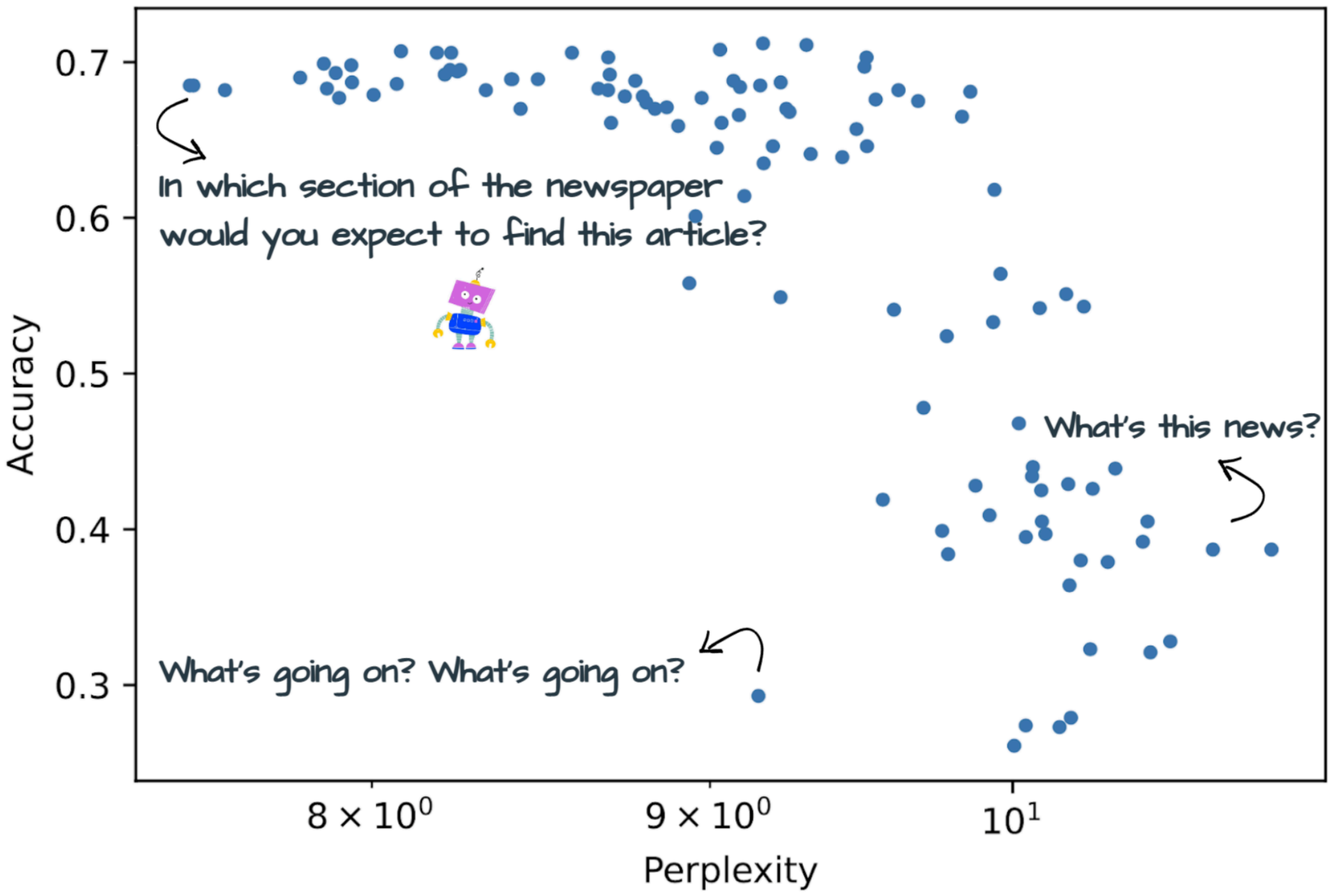
- **Approach 1:** Generate from the model and read off the generation
  - What if you ask for a star rating and it doesn't give you a number of stars but just says something else?
- **Approach 2:** Compare probs: "*Out of positive, negative, or neutral, this review is _*" Compare P(*positive* | context), P(*neutral* | context), P(*negative* | context)
  - This constrains the model to only output a valid answer, and you can normalize these probabilities to get a distribution

Variability in Prompts

- Plot: large number of prompts produced by {manual writing, paraphrasing, backtranslation}

- A little prompt engineering will get you somewhere decent

y-axis: task performance



x-axis: perplexity of the prompt. How natural is it? How much does it appear in the pre-training data?

Gonen et al. (2022)

Variability in Prompts

- OPT-175B: average of best 50% of prompts is much better than average over all prompts

Task	Avg Acc	Acc 50%
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

Gonen et al. (2022)

Prompt Optimization

- A number of methods exist for searching over prompts (either using gradients or black-box optimization)
- Most of these do not lead to dramatically better results than doing some manual engineering/hill-climbing (and they may be computationally intensive)
- Nevertheless, the choice of prompt *is* very important in general for zero-shot settings! We will see more next time.
- In two lectures: models that are trained to do better at prompts (RLHF)

Few-shot Prompting

Few-shot Prompting

- Form “training examples” from (**x**, y) pairs, verbalize them (can be lighter-weight than zero-shot verbalizer)
- Input to GPT-3: $\boldsymbol{v}(\boldsymbol{x}_1)\ \boldsymbol{v}(y_1)\ \boldsymbol{v}(\boldsymbol{x}_2)\ \boldsymbol{v}(y_2)\ \ldots\ \boldsymbol{v}(\boldsymbol{x}_{\text{test}})$

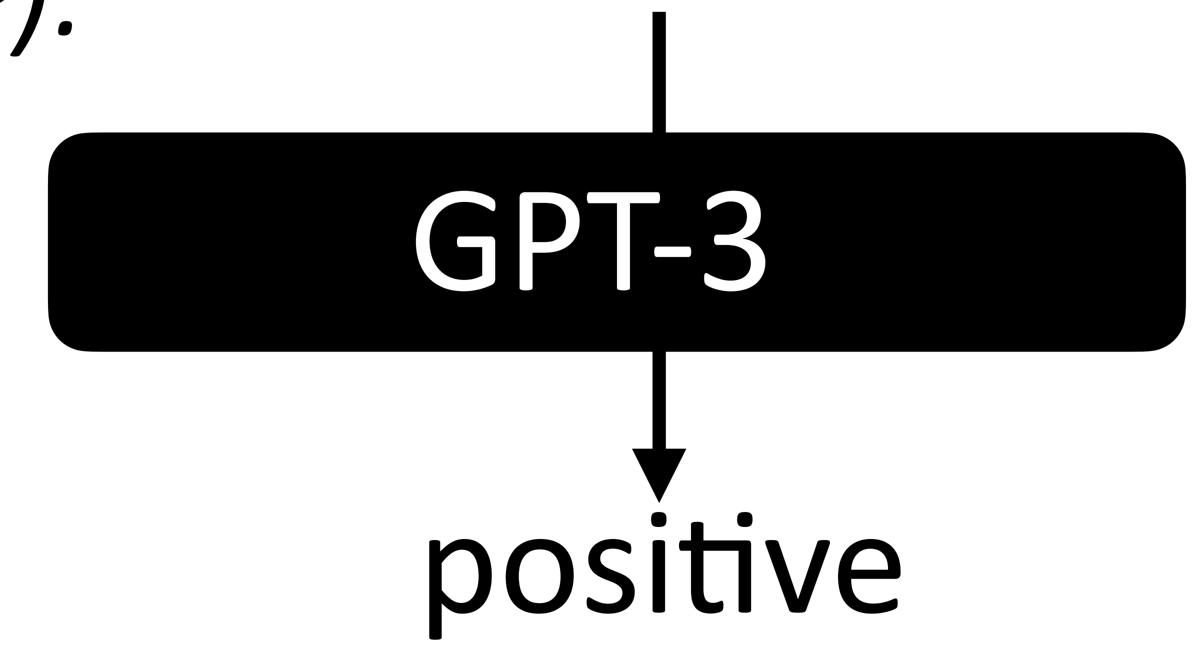
Review: The cinematography was stellar; great movie!
Sentiment (positive or negative): positive

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):



What can go wrong?

Review: The movie was great!
Sentiment: positive

Review: I thought the movie was alright; I would've seen it again.

Sentiment: positive

Review: The movie was pretty cool!

Sentiment: positive

Review: Pretty decent movie!

Sentiment: positive

Review: The movie had good enough acting and the visuals were nice.

Sentiment: positive

Review: There wasn't anything the movie could've done better.

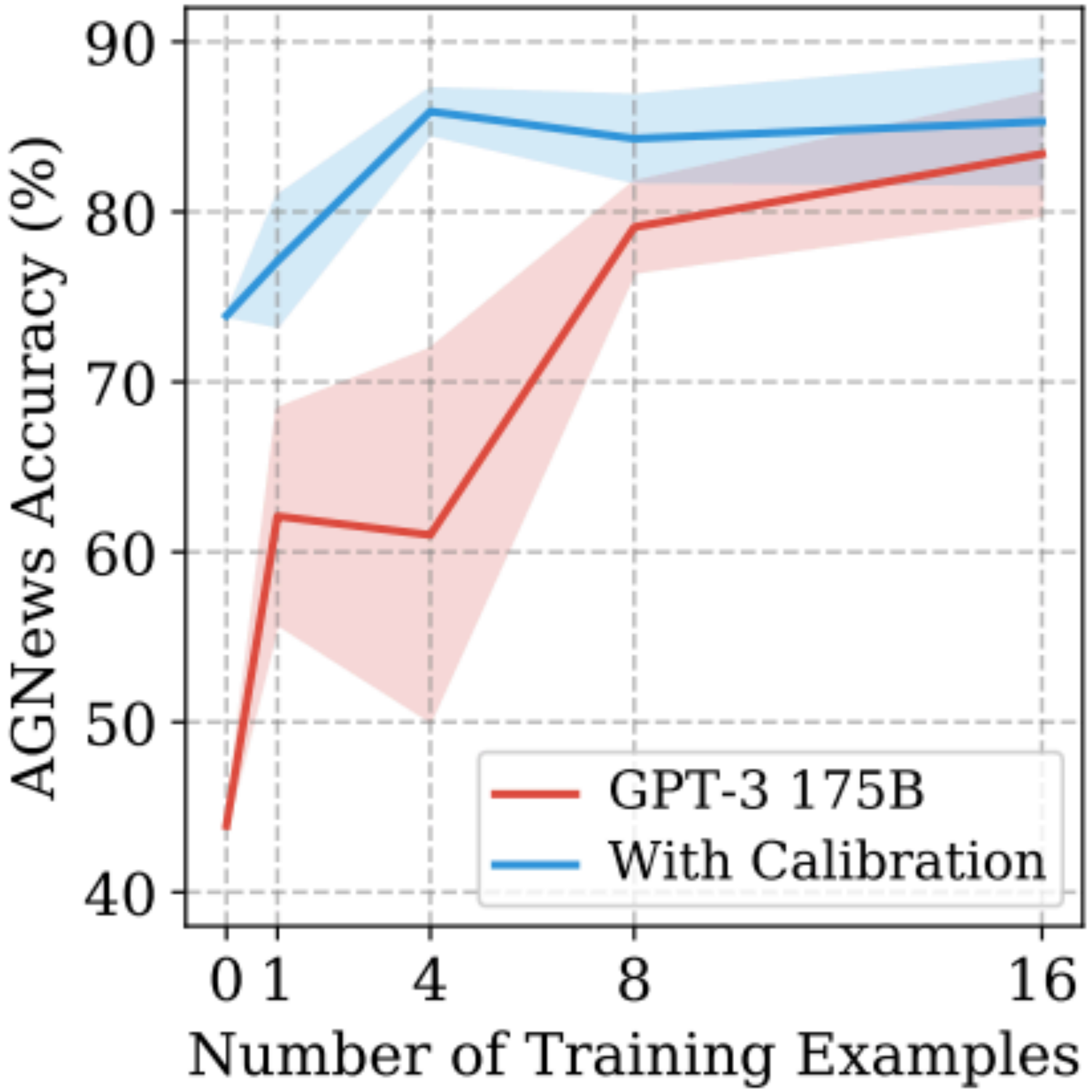
Sentiment: positive

Review: Okay movie but could've been better.
Sentiment: → GPT-3 → positive

What can go wrong?

- What if we take random sets of training examples? There is quite a bit of variance on basic classification tasks, due to effects like this

- Note: these results are with basic GPT-3 and not Instruct-tuned versions of the model. This issue has gotten a lot better

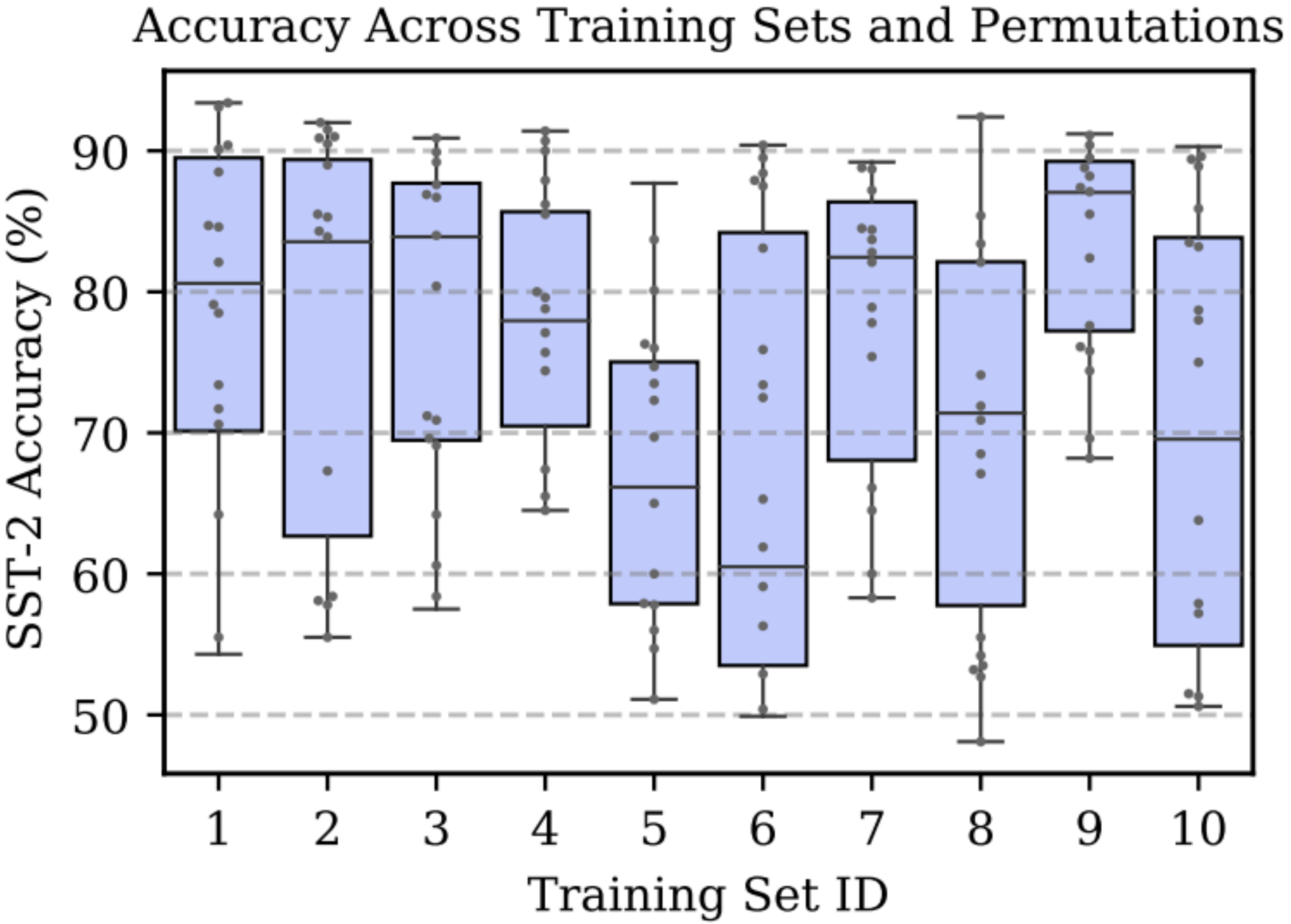


Zhao et al. (2021)

What can go wrong?

- Varies even across permutations of training examples

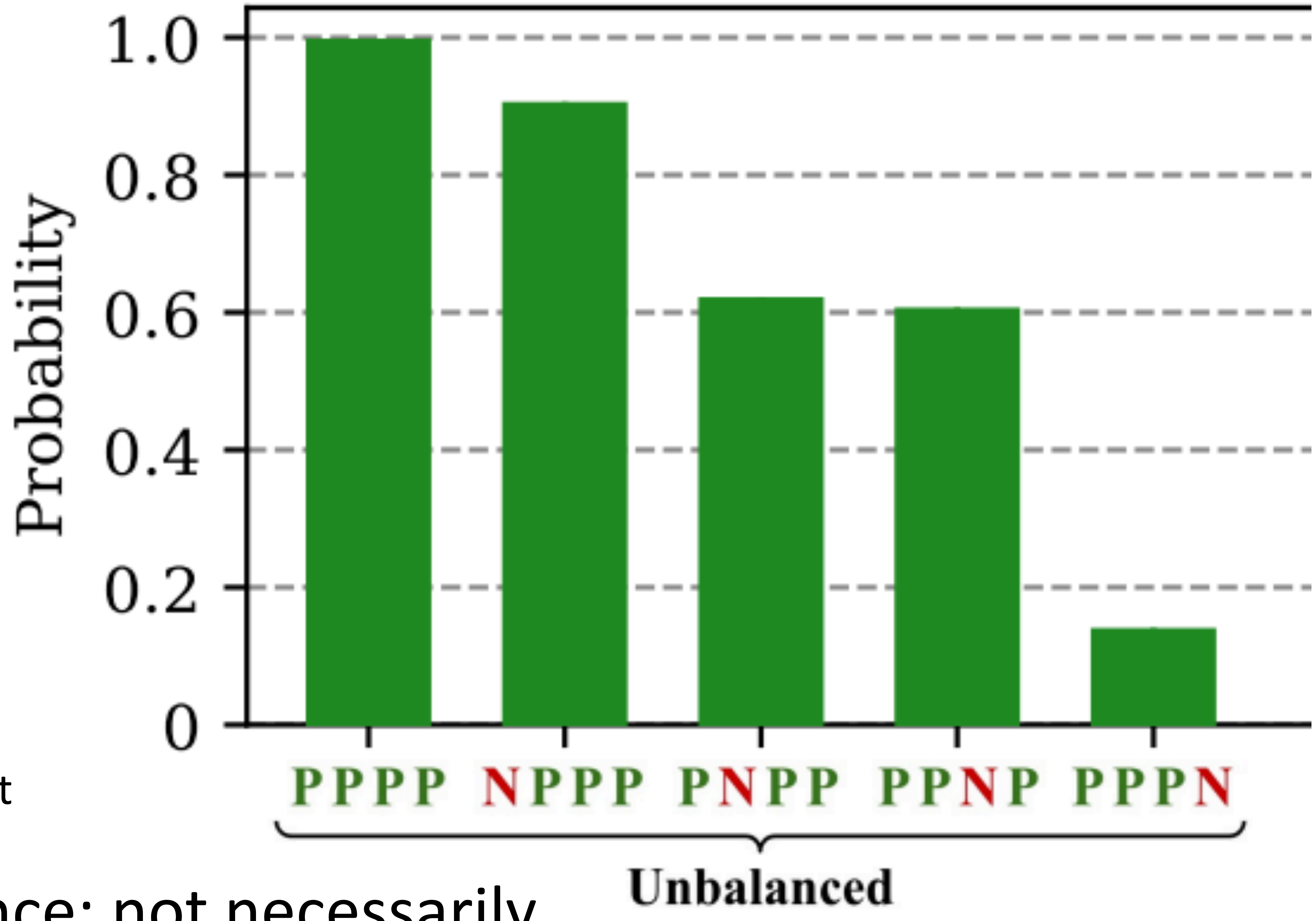
- x-axis: different collections of train examples.
  y-axis: sentiment accuracy. Boxes represent results over different permutations of the data



Zhao et al. (2021)

What can go wrong?

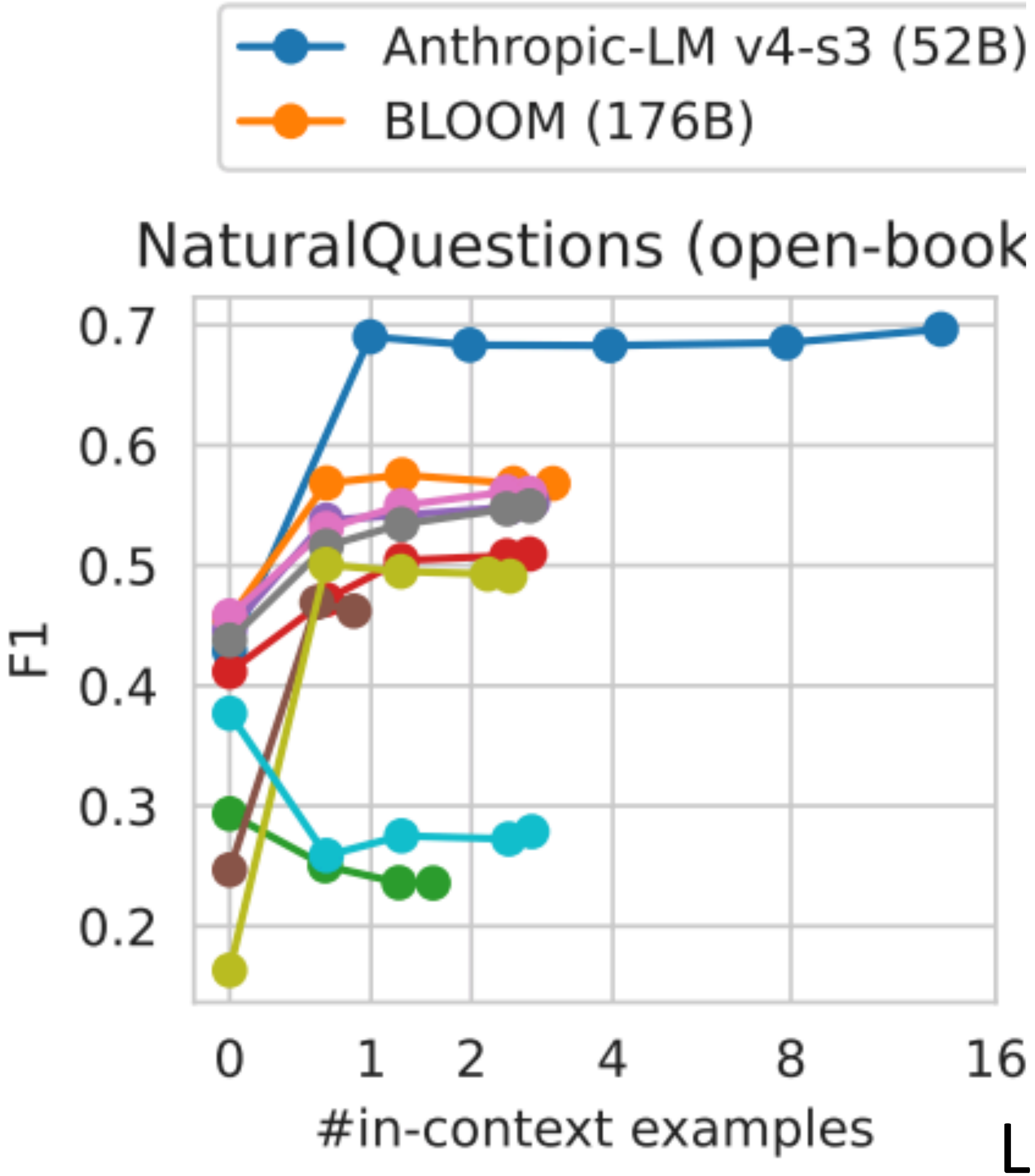
- Having unbalanced training sets leads to high "default" probabilities of positive; that is, if we feed in a null $\boldsymbol{x}_{\text{test}}$
- Solution: "calibrate" the model by normalizing by that probability of null $\boldsymbol{x}_{\text{test}}$
- Leads to higher performance; not necessarily crucial with prompt-tuned models



Zhao et al. (2021)

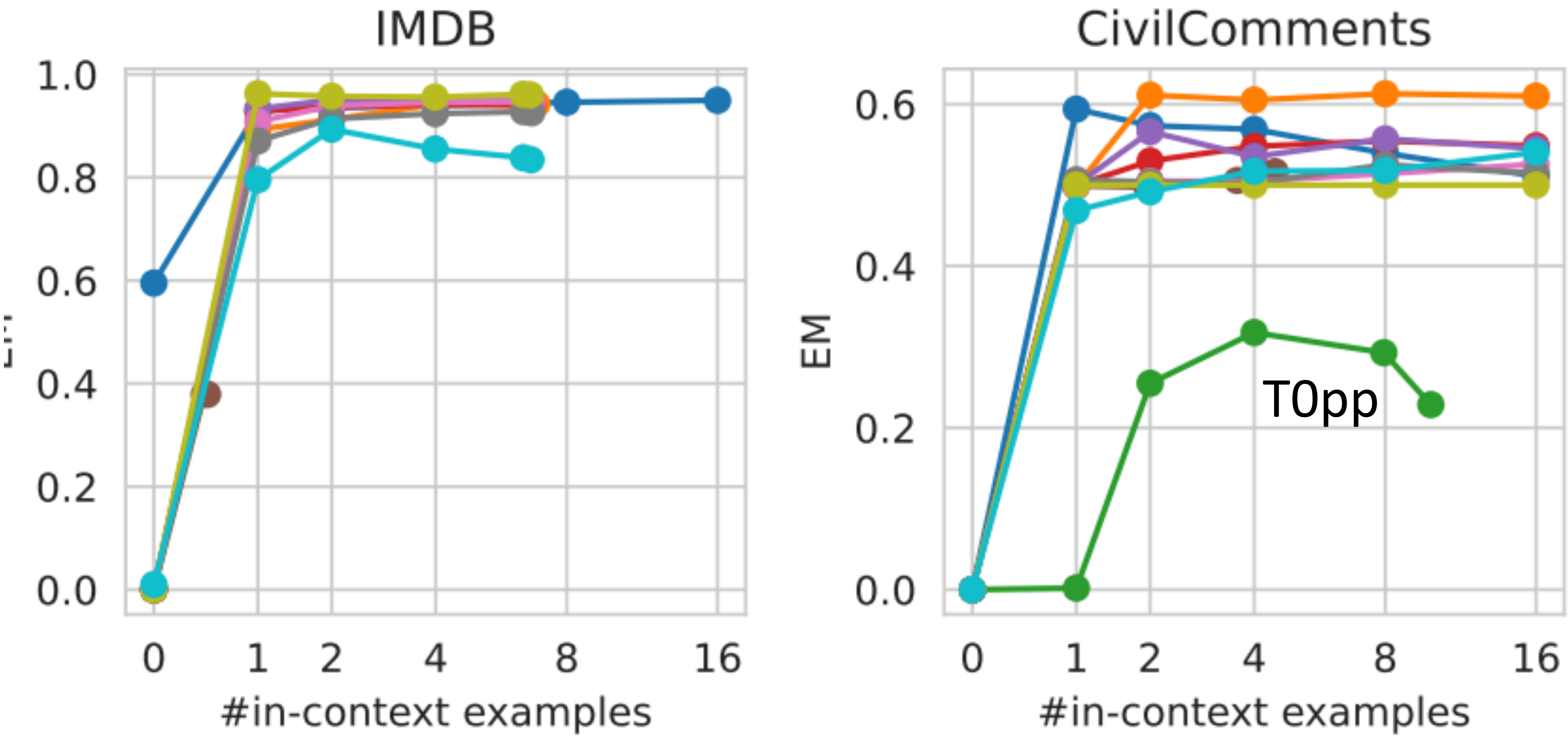
Results: HELM

- So, how much better is few-shot compared to zero-shot?
- Each line is a different LM
- More in-context examples generally leads to better performance
- What do we see here?



Liang et al. (2022)

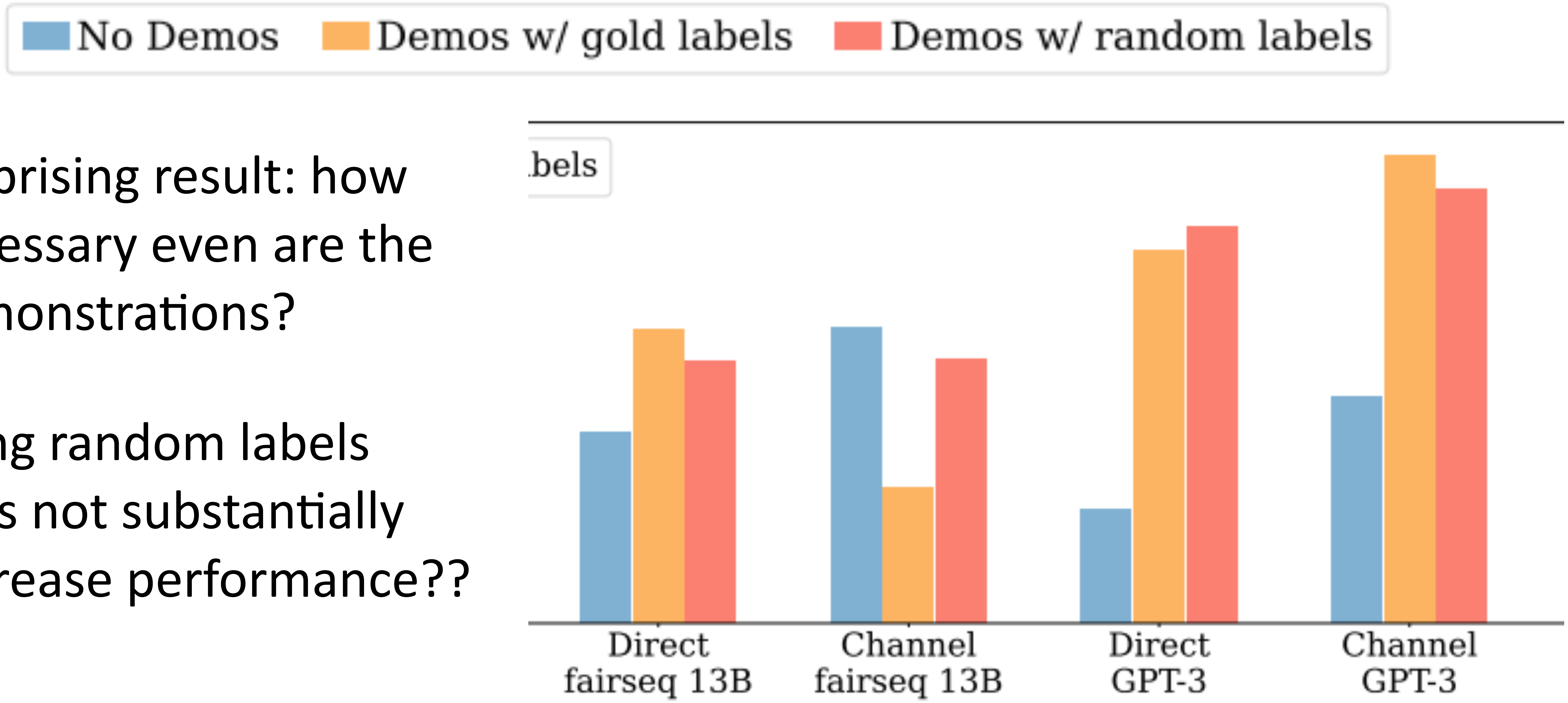
Results: HELM



- What trends do these show?

Liang et al. (2022)

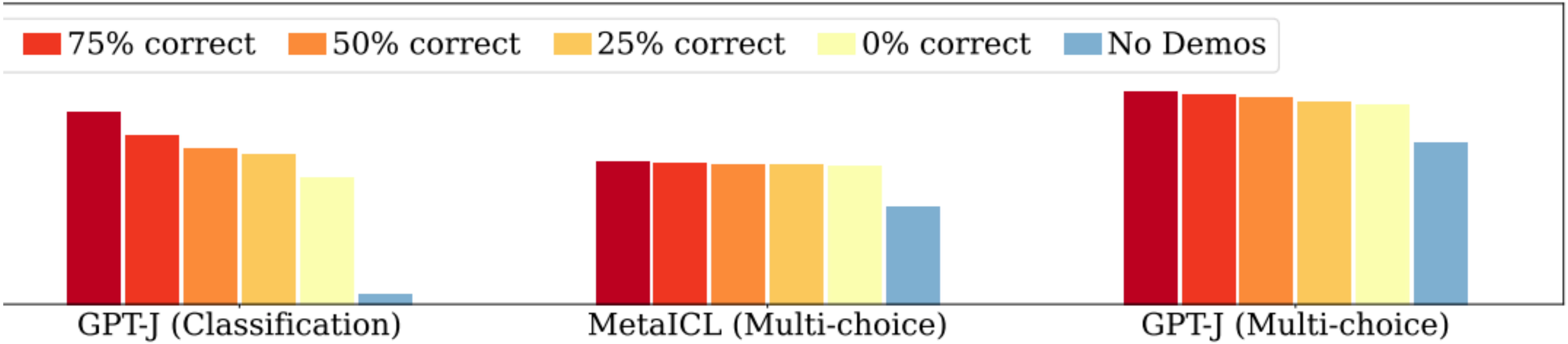
Rethinking Demonstrations



- Surprising result: how necessary even are the demonstrations?
- Using random labels does not substantially decrease performance??

Min et al. (2022)

Rethinking Demonstrations



- Having even mislabeled demonstrations is much better than having no demonstrations, indicating that the form of the demonstrations is partially responsible for in-context learning

Min et al. (2022)

Factuality and Hallucination

Factuality

- When you fine-tune a bag-of-words model on sentiment, you learn word meanings *from the data itself*

- When you fine-tune an embedding-based model or BERT on sentiment, you still learn from the data, and the pre-training helps generalize

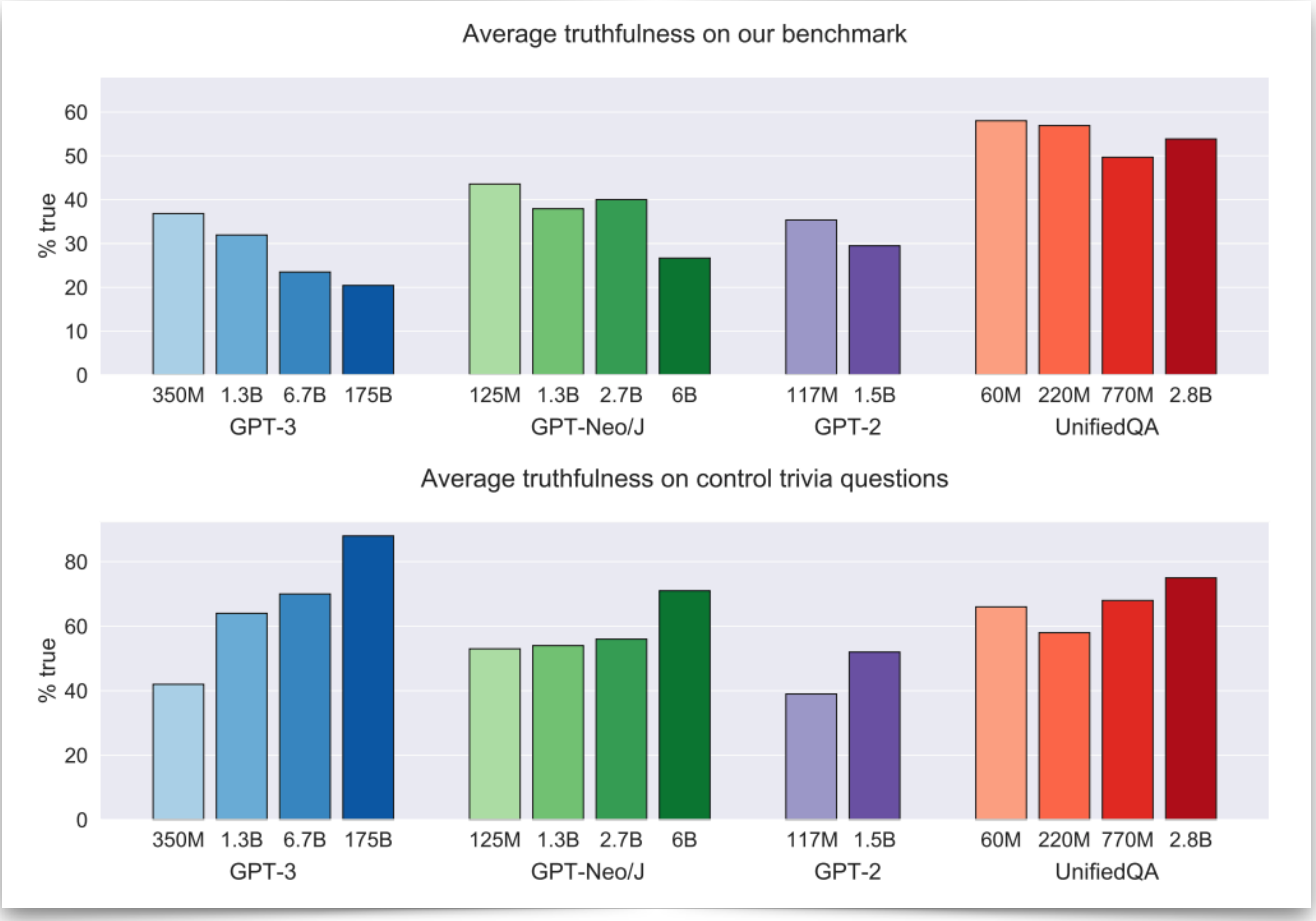
- When a language model is **prompted** to do a task like sentiment, you really don't see enough data points to "learn" much. You're relying on the model's pre-training

- What implications does this have for producing factual knowledge from LMs?

Factuality

- Language models model distributions over text, not facts. There's no guarantee that what they generate is factual:
  - Language models are trained on the web. Widely-popularized falsehoods may be reproduced in language models
  - A language model may not be able to store all rare facts, and as a result moderate probability is assigned to several options

TruthfulQA


Average truthfulness on our benchmark
% true
0
10
20
30
40
50
60
350M
1.3B
6.7B
175B
GPT-3
125M
1.3B
2.7B
6B
GPT-Neo/J
117M
1.5B
GPT-2
60M
220M
770M
2.8B
UnifiedQA
Average truthfulness on control trivia questions
% true
0
20
40
60
80
350M
1.3B
6.7B
175B
GPT-3
125M
1.3B
2.7B
6B
GPT-Neo/J
117M
1.5B
GPT-2
60M
220M
770M
2.8B
UnifiedQA

Factuality

- Language models model distributions over text, not facts. There's no guarantee that what they generate is factual:
  - Language models are trained on the web. Widely-popularized falsehoods may be reproduced in language models
- A language model may not be able to store all rare facts, and as a result moderate probability is assigned to several options
- There are many proposed solutions to factuality. How do we evaluate them? How do we check facts "explicitly"?

Grounding LM Generations

- Suppose we have text generated from an LM. We want to check it against a source document. What techniques have we seen so far that can do this?

- What steps are involved?

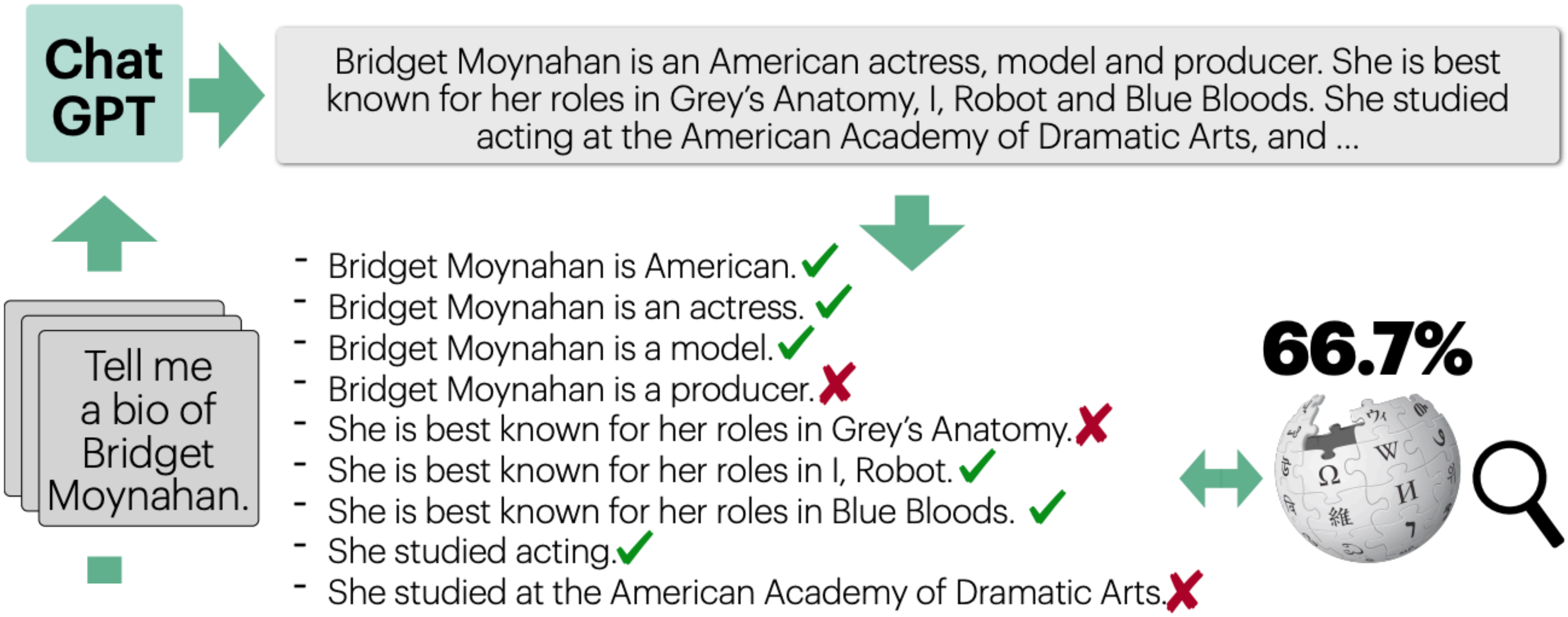
  1. Decide what text you are grounding in (may involve retrieval)

  2. Decompose your text into pieces of meaning to ground

  3. Check each piece

- For now, we'll assume the reference text/documents are given to us and not focus on step 1

Concrete Setting

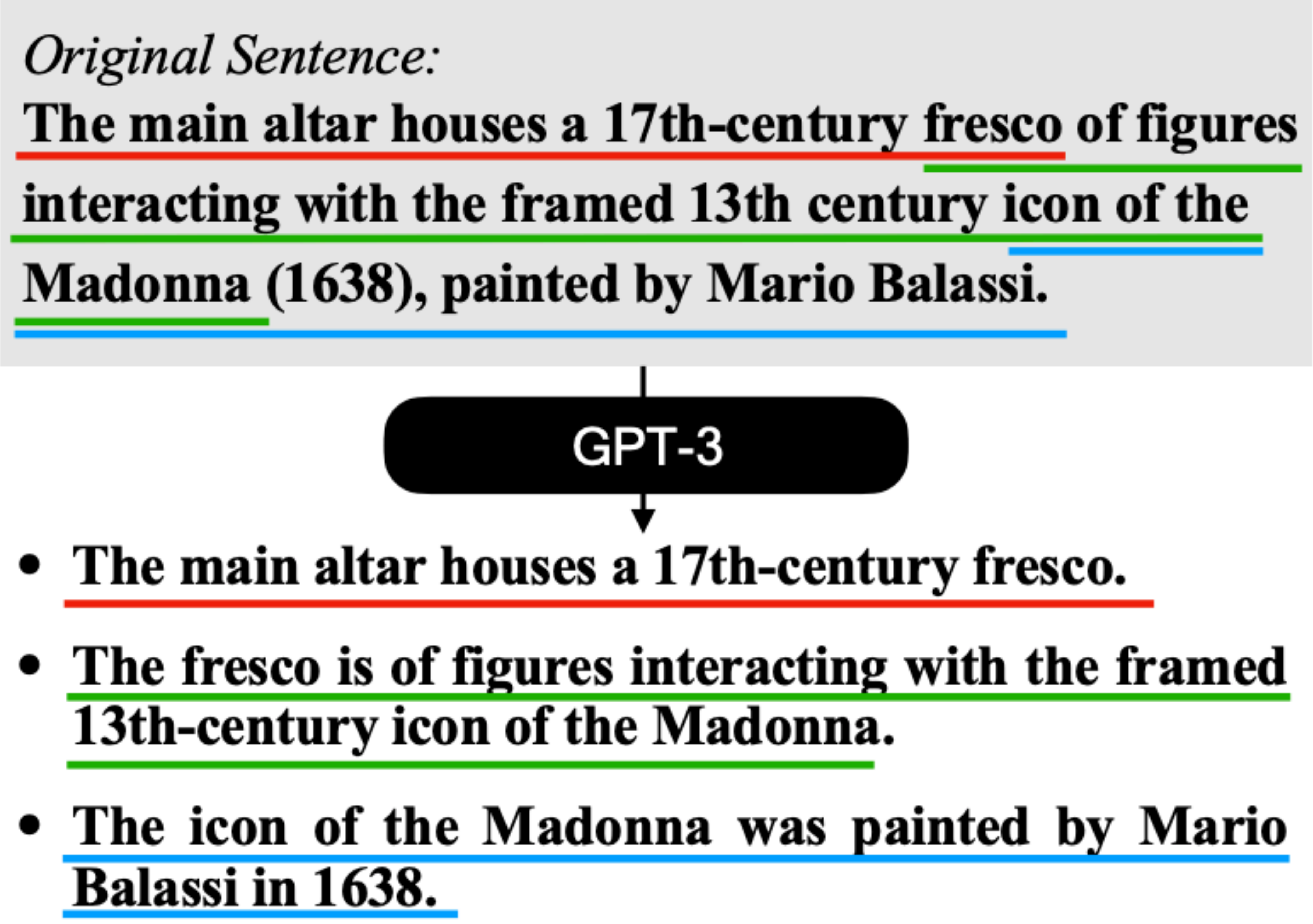


- Dataset: ChatGPT-generated biographies of people. May contain errors, particularly when dealing with obscure people!

Sewon Min and Kalpesh Krishna et al. (2023)

Step 2: Decomposition

- Simplest approach: each sentence needs to be grounded
- Can go deeper: think of sentences as expressing a collection of propositions
- Long history in frame semantics of defining these propositions. Many propositions anchor to verbs
- Recent work: extract propositions with LLMs

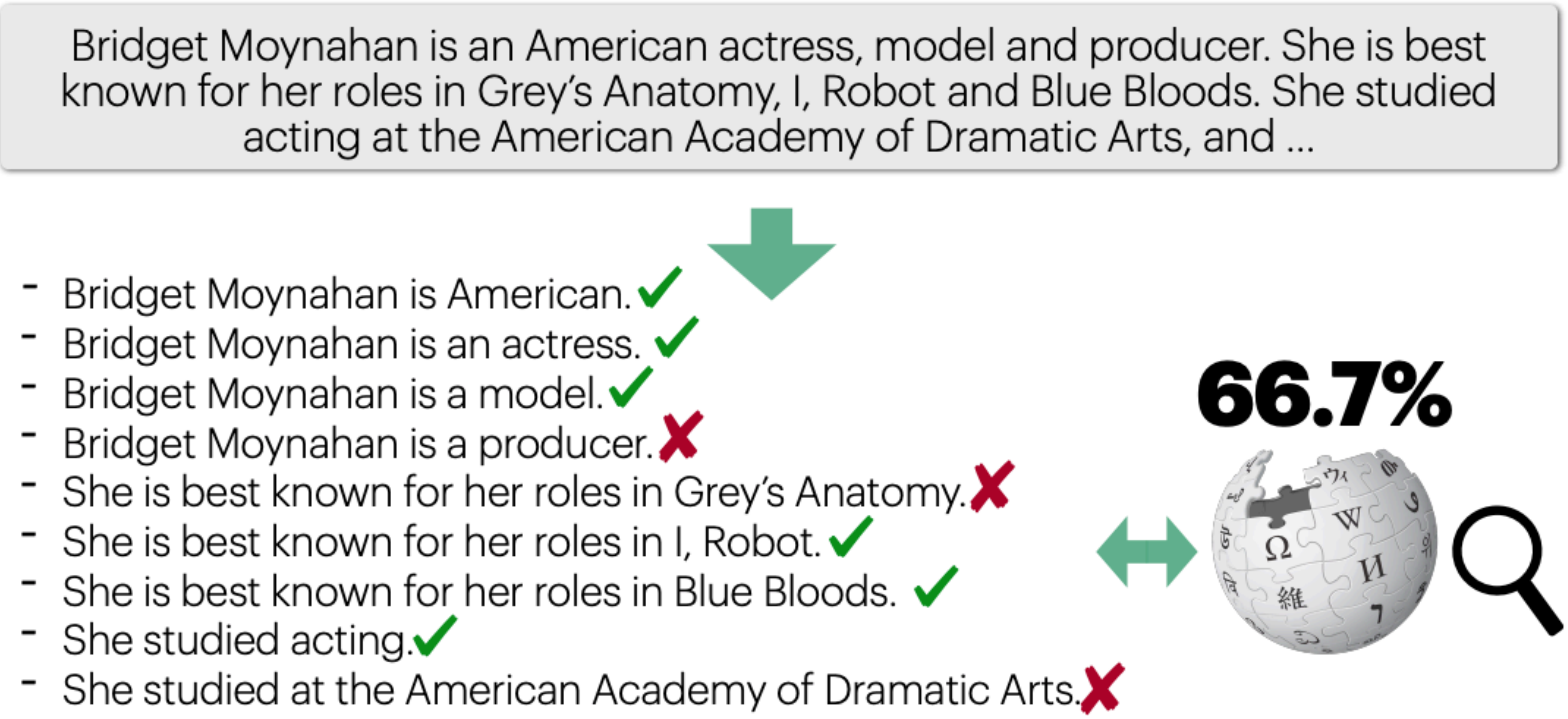


Yixin Liu et al. (2023)

Ryo Kamoi et al. (2023)

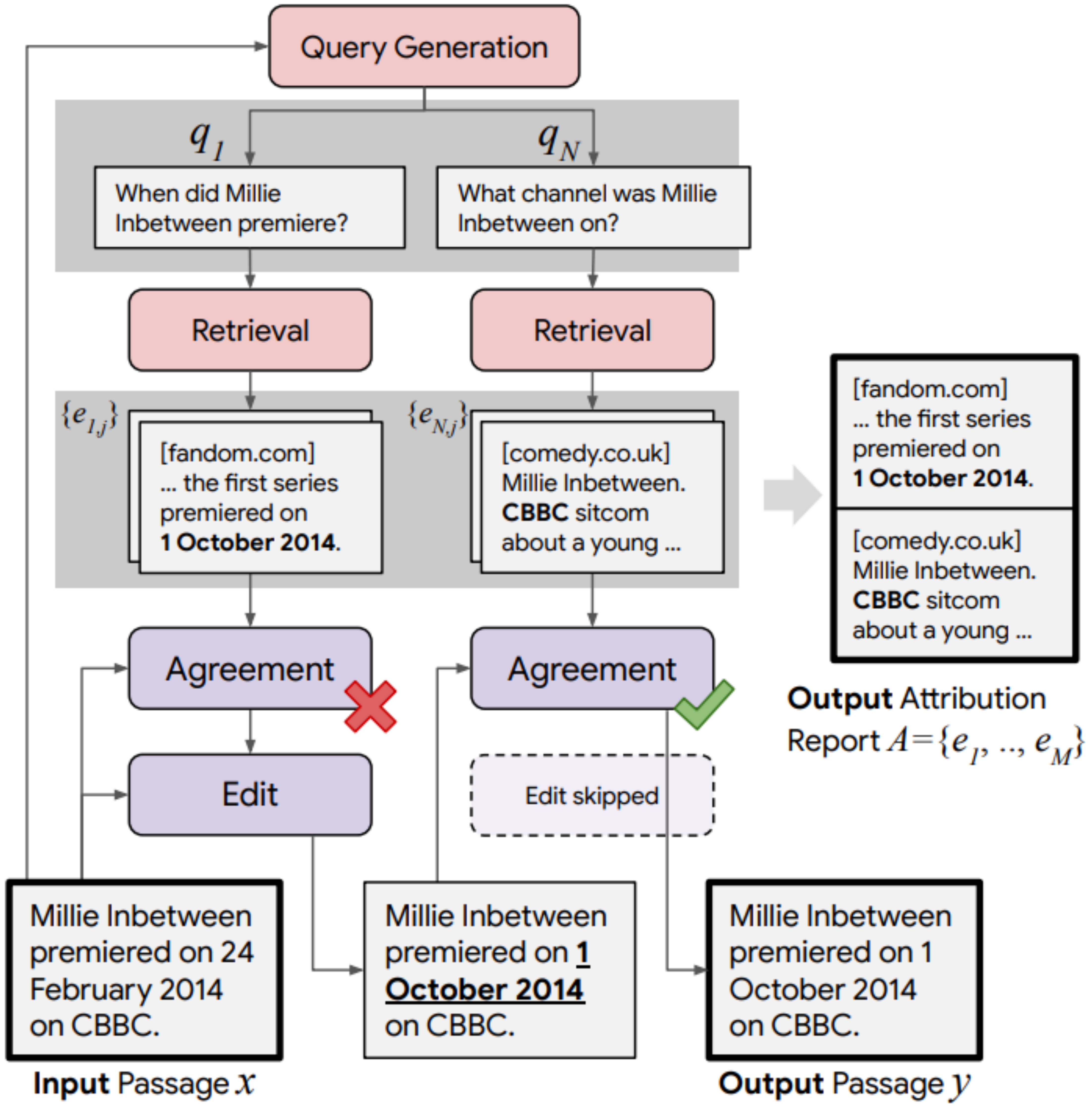
Assignment 5

- Your task: look at how to verify these facts against passages from Wikipedia
- You'll look at two methods: word overlap and entailment models (from Hugging Face)
- Error analysis: are the facts right? Do the retrieved documents support them?



Sewon Min and Kalpesh Krishna et al. (2023)

Pipeline: RARR



- Full pipeline including retrieval
- Decomposition is framed as question generation
- The "checking" stage is also implemented with LLMs here
- Final stage: try to revise the output

Luyu Gao et al. (2022)

Understanding ICL: Induction Heads and Mechanistic Interpretability

Background: Transformer Circuits

- There are mechanisms in Transformers to do “fuzzy” or “nearest neighbor” versions of pattern completion, completing [A*][B*] … [A] → [B] , where A* ≈ A and B* ≈ B are similar in some space

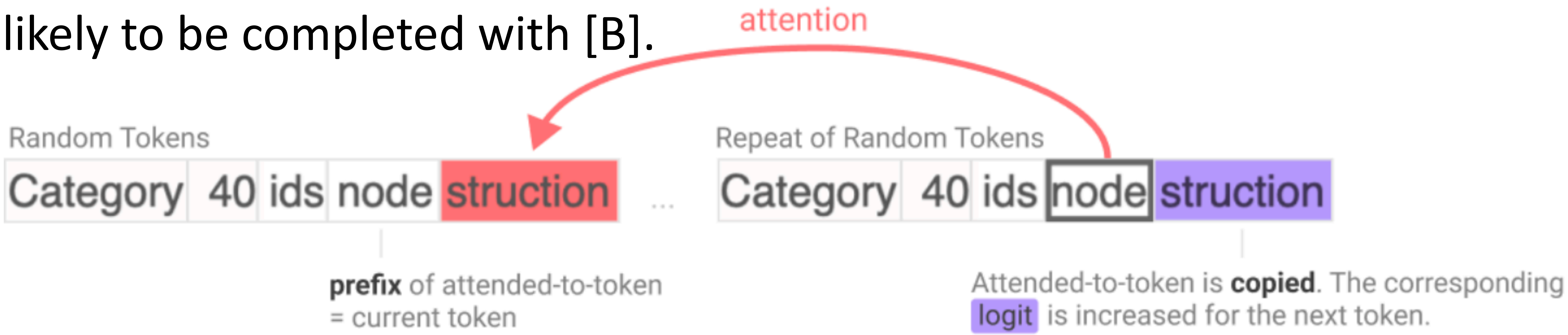
- Olsson et al. want to establish that these mechanisms are responsible for good ICL capabilities

- We can find these heads and see that performance improves; can we causally link these?

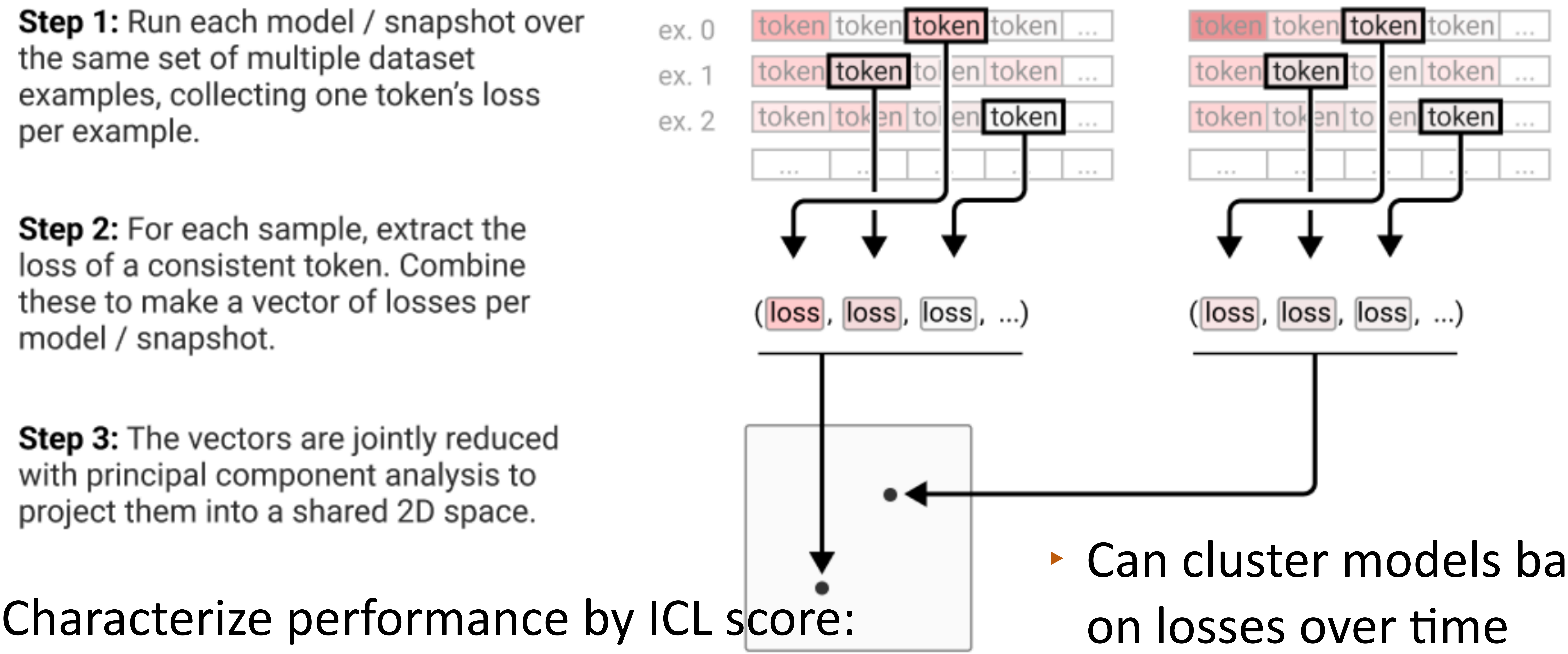
Olsson et al. (2022)

Induction Heads

- Induction heads: a pair of attention heads in different layers that work together to copy or complete patterns.
- The first head copies information from the previous token into each token.
- Second attention head to attend to tokens based on what happened before them, rather than their own content. Likely to “look back” and copy next token from earlier
- The two heads working together cause the sequence …[A][B]…[A] to be more likely to be completed with [B].



Induction Heads



- Characterize performance by ICL score: loss(500th token) - loss(50th token) — average measure of how much better the model is doing later once it's seen more of the pattern

- Can cluster models based on losses over time

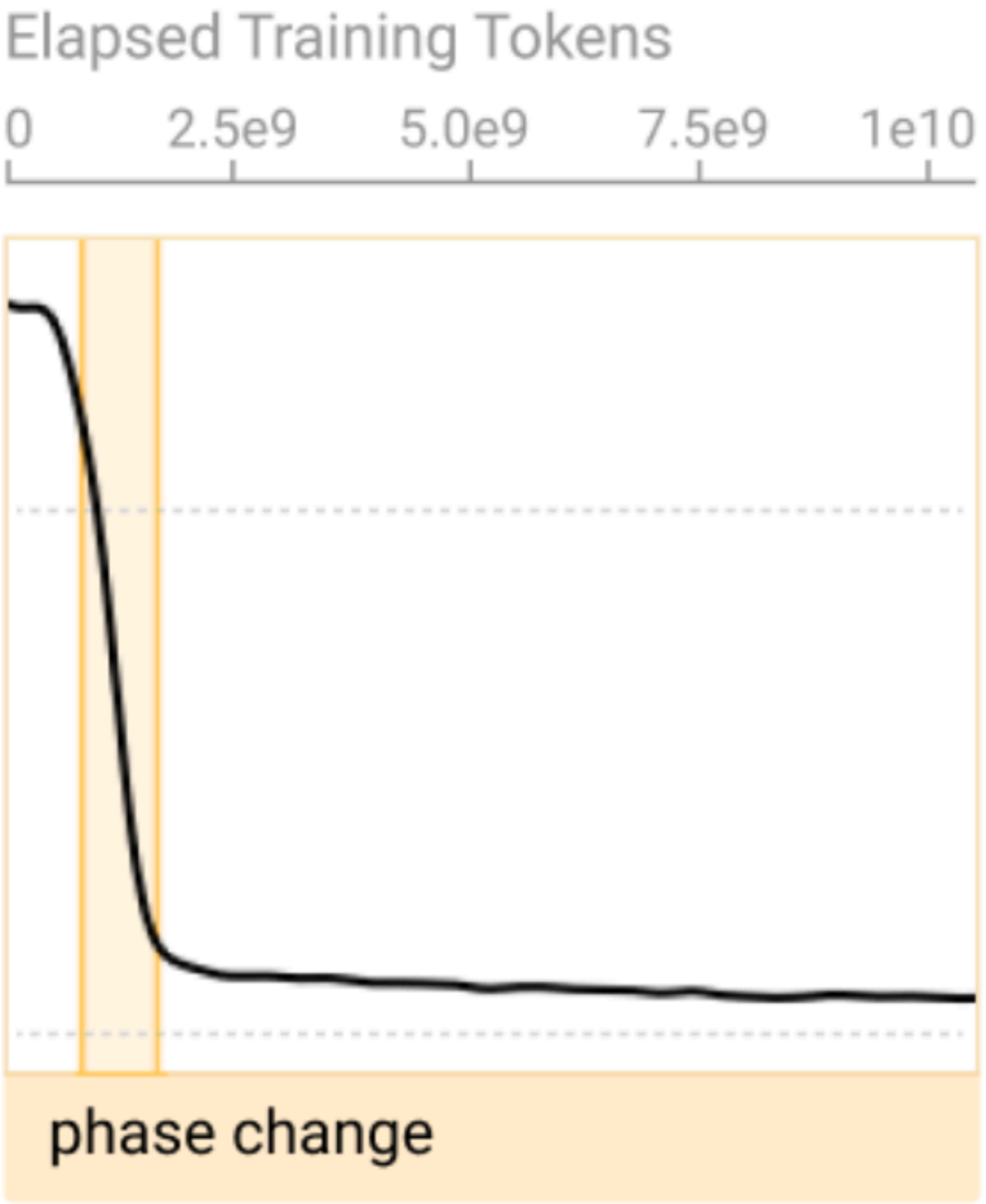
Olsson et al. (2022)

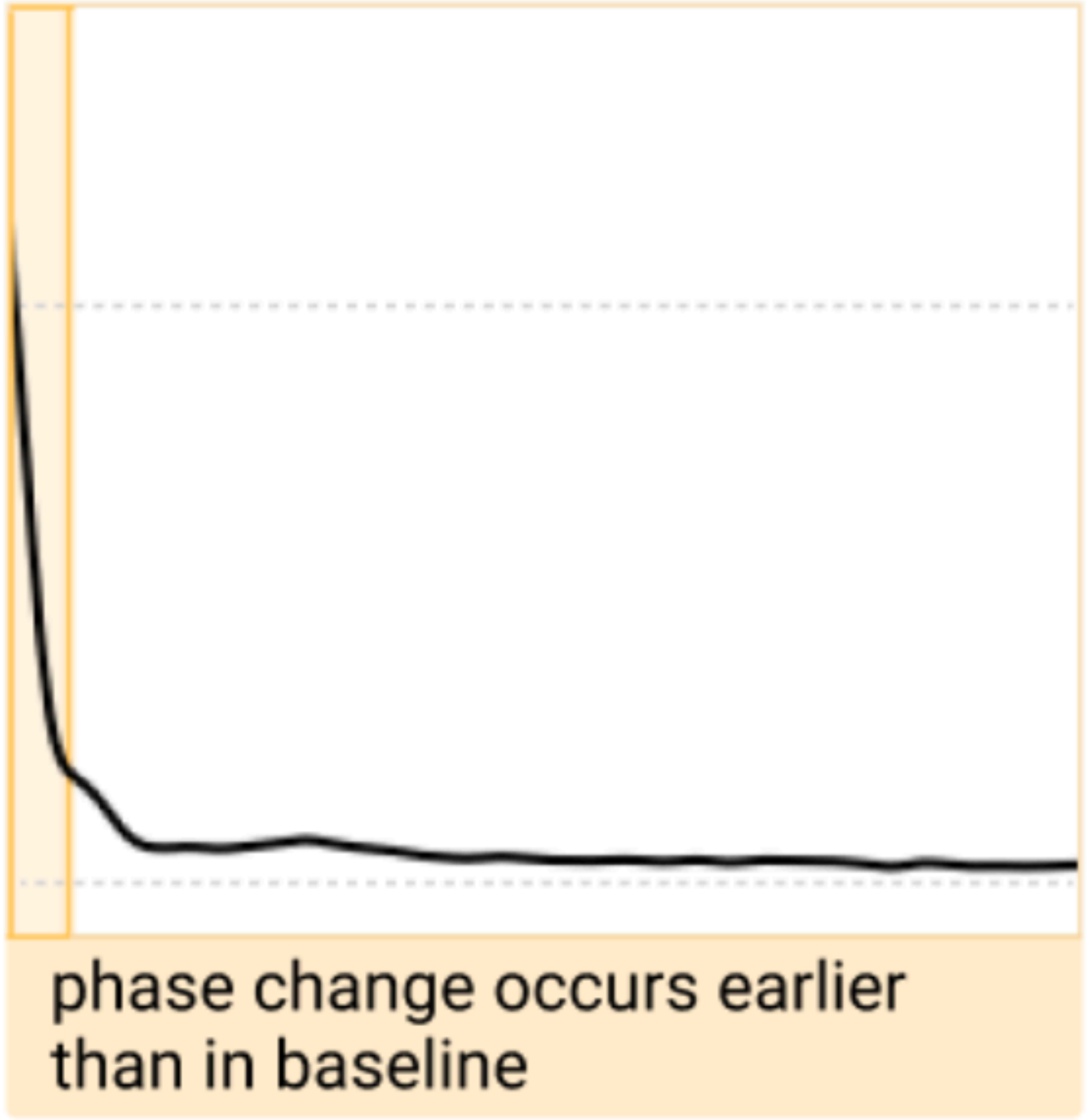
Induction Heads

- Improvement in ICL (loss score) correlates with emergence of induction heads

Induction Heads

Change architecture to promote induction heads => phase change happens earlier





Induction Heads

- If you remove induction heads, behavior changes dramatically

Interpretability

- Lots of explanations for why ICL works — but these haven't led to many changes in how Transformers are built or scaled

- Several avenues of inquiry: theoretical results (capability of these models), mechanistic interpretability, fully empirical (more like that next time)

- Many of these comparisons focus on GPT-3 and may not always generalize to other models

Takeaways

- Zero- and few-shot prompting are very powerful ways of specifying new tasks at inference time
- For zero-shot: form of the prompt matters, we'll see more example next times when we look at chain-of-thought
- For few-shot: number and order of the examples matters, prompt matters a bit less
- Several analyses of why it works: it can learn to do regression and we know a bit about mechanisms that may be responsible for it