

# Understanding and Improving Pre-trained Large Language Models through a Probabilistic Lens

Xinyi Wang

# Language Model

- **Definition:** a probability distribution  $P$  over sequences of words  $w_1, w_2, \dots, w_T$ .
- Different assumptions on decomposing this joint probability produce different types of language models.

Classic  
language  
models:

Bag of words model

$$p(w_1, w_2, \dots, w_T) = \prod_{i=1}^T p(w_i)$$

Hidden Markov model

$$p(w_1, w_2, \dots, w_T) = \sum_{h_0, h_1, \dots, h_T \in H} p(h_0) \prod_{i=1}^T p(w_i | h_i) p(h_i | h_{i-1})$$

Neural  
language  
models:

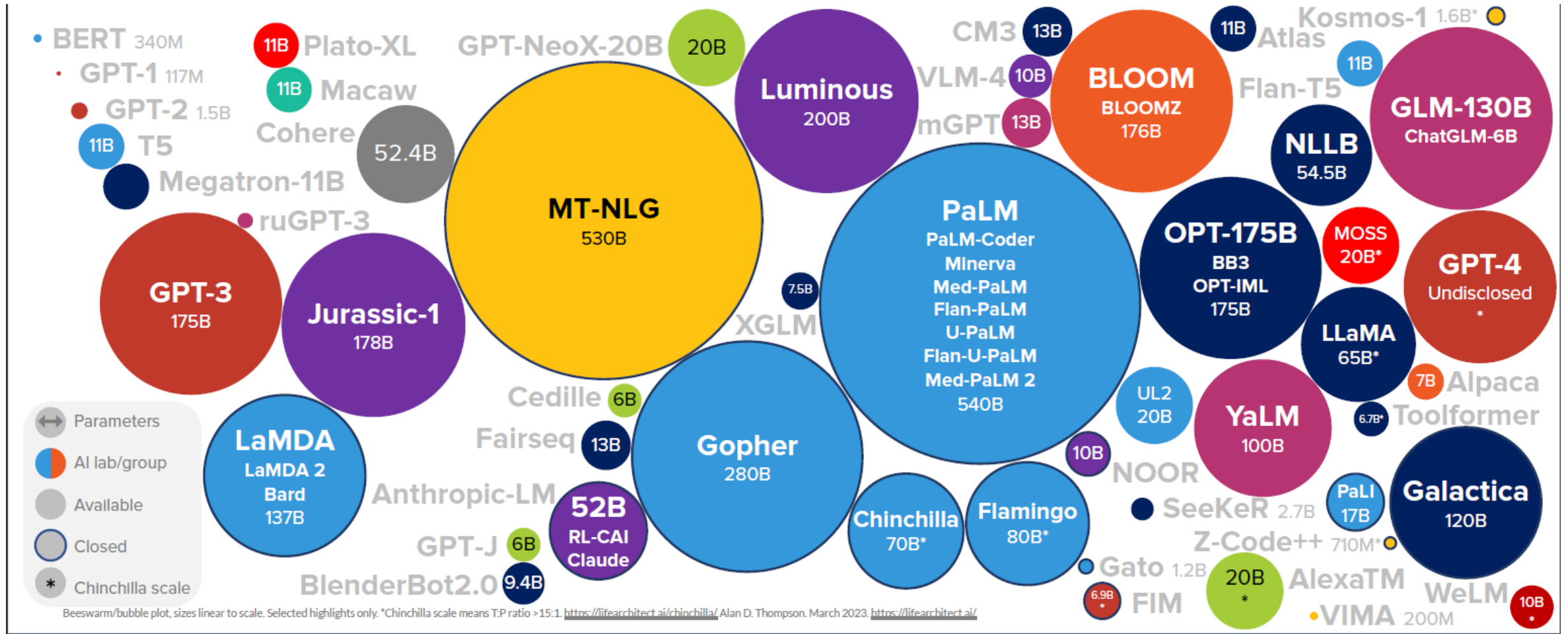
Generative language model

$$p(w_1, w_2, \dots, w_T) = \prod_{i=1}^T p(w_i | w_1, w_2, \dots, w_{i-1})$$

Masked language model

$$p(w_1, w_2, \dots, w_T) = \prod_{i=1}^T p(w_i | w_1, \dots, w_{i-1}, w_{i+1}, \dots, w_T)$$

# Existing large language models



- Real-world exponential parameter growth of large language models ([source](#)).

# Curious capabilities of LLMs

- **Fine-tuning:** pre-trained LLMs are good starting points for downstream tasks.
- Prompting: use a LLM as it is. Usually require instruction tuning.
  - **In-context learning** ([Brown et al., 2020](#))
  - **chain-of-thoughts prompting** ([Wei et al., 2022](#))
  - ...
- Scaling:
  - **Exponential scaling law:** test loss v.s. model size, dataset size, compute ([Kaplan et al., 2020](#))
  - **Emergent abilities:** An ability is emergent if it is not present in smaller models but is present in larger models. ([Wei et al., 2022](#))

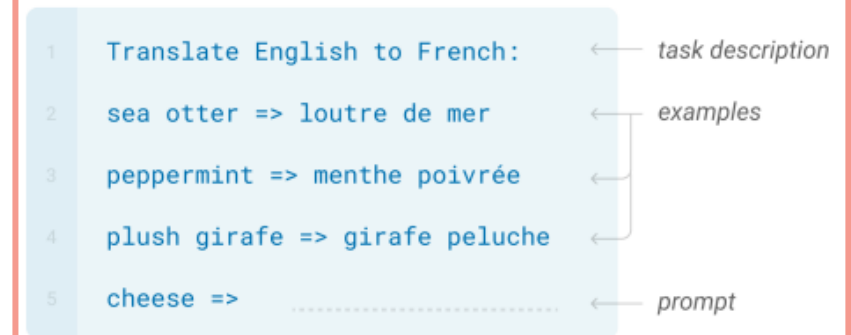
## Zero-shot

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



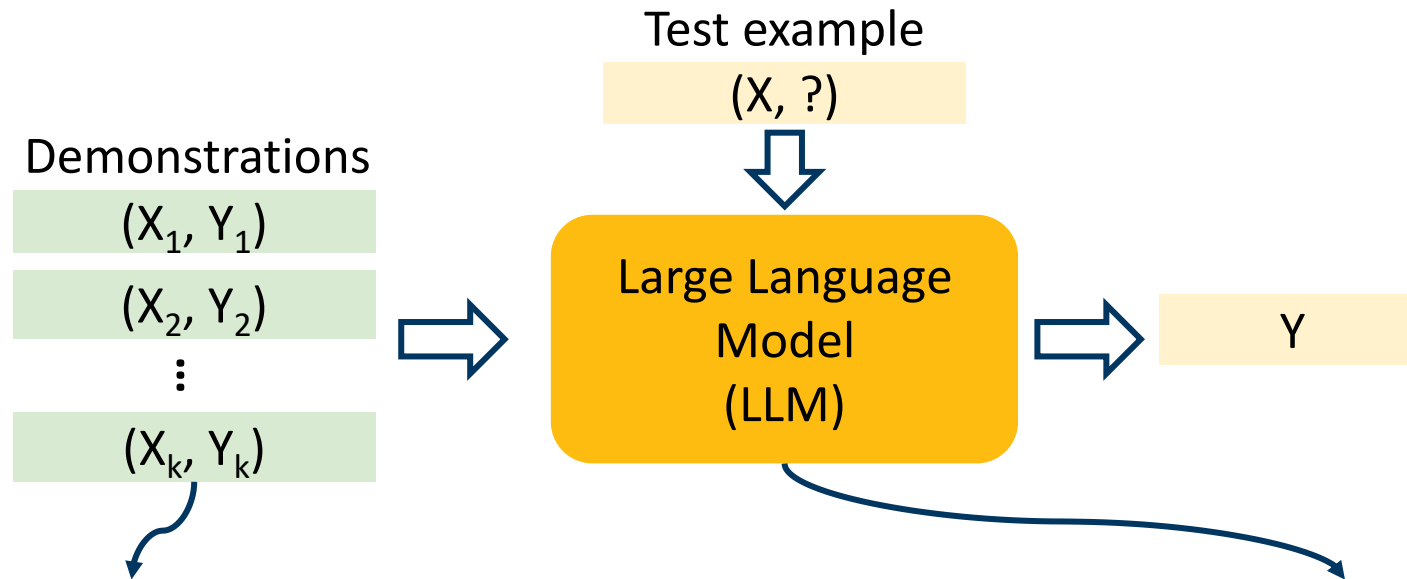
## Few-shot

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



([Brown et al., 2020](#))

# How to understand in-context learning?



In-context learning is highly **unstable**:

- How to choose a set of demonstrations if we have some annotated data? Similarity? (Liu et al. 2022; Su et al. 2022) Entropy of predicted labels? (Lu et al. 2022)

Why LLMs can do in-context learning:

- Pretraining distribution? HMM (Xie et al., 2022)? Long tailed? Burstiness (Chan et al. 2022)?
- Mimicking gradient descent? (von Oswald et al. 2022, Akyurek et al. 2022, [Dai et al. 2023](#))

UC SANTA BARBARA

University of California, Irvine

# Large Language Models are Implicitly Topic Models: Explaining and Finding Good Demonstrations for In- Context Learning

Xinyi Wang, Wanrong Zhu, Michael Saxon, Mark Steyvers, William Yang  
Wang ([NeurIPS 2023](#))

# LLMs are implicitly topic models

$$\text{LLM: } P(w_{1:T}) = \prod_{i=1}^T P(w_i | w_{i-1}, \dots, w_1) \quad \text{Topic model: } P(w_{1:T}) = \int_{\Theta} P(w_{1:T} | \theta) P(\theta) d\theta$$

Language model probability output by an LLM

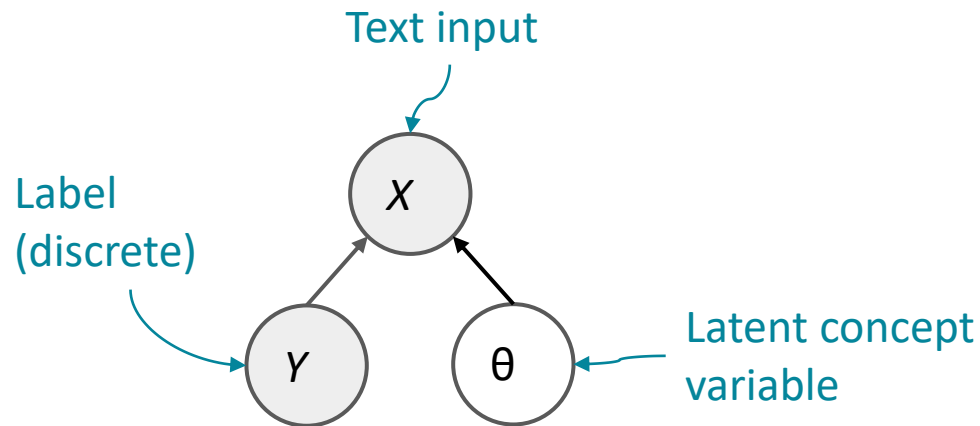
**Our assumption:**  $P_M(w_{t+1:T} | w_{1:t}) = \int_{\Theta} P_M(w_{t+1:T} | \theta) P_M(\theta | w_{1:t}) d\theta$

Generated continuation      Prompt      LLMs generate the continuation exclusively based on the inferred concept variable  $\theta$       LLMs implicitly infer a latent concept variable  $\theta$  from the prompt

The diagram illustrates the components of the equation  $P_M(w_{t+1:T} | w_{1:t}) = \int_{\Theta} P_M(w_{t+1:T} | \theta) P_M(\theta | w_{1:t}) d\theta$ . A blue arrow points from the text 'Language model probability output by an LLM' to the circled  $P_M$  in the equation. Another blue arrow points from 'Generated continuation' to the  $w_{t+1:T}$  term. A third blue arrow points from 'Prompt' to the  $w_{1:t}$  term. A fourth blue arrow points from 'LLMs generate the continuation exclusively based on the inferred concept variable  $\theta$ ' to the  $P_M(w_{t+1:T} | \theta)$  term. A fifth blue arrow points from 'LLMs implicitly infer a latent concept variable  $\theta$  from the prompt' to the  $P_M(\theta | w_{1:t})$  term. Red underlines are placed under  $w_{t+1:T}$ ,  $w_{1:t}$ ,  $P_M(w_{t+1:T} | \theta)$ , and  $P_M(\theta | w_{1:t})$ .

- Assumption: the generated continuation is independent of the prompt given the concept variable  $\theta$ .

# Data generation direction matters

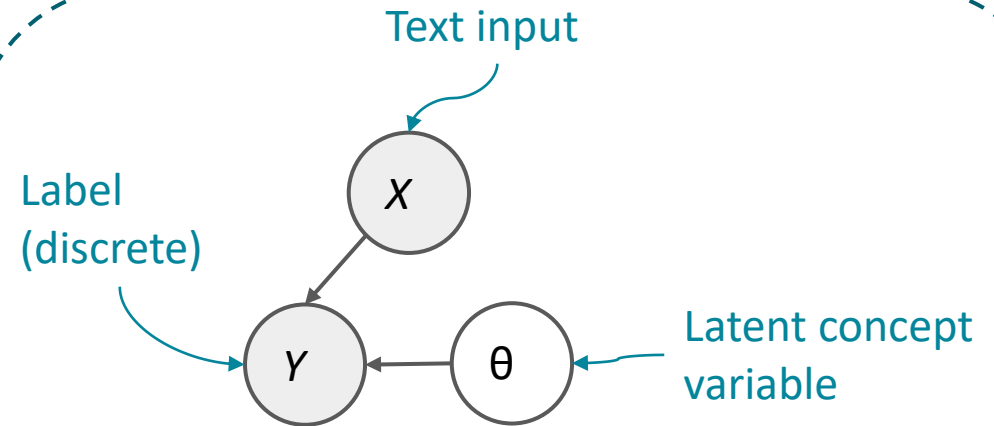


e.g. sentiment analysis, topic classification, emotion classification tasks

$$P_M^d(X|Y_1^d, X_1^d, \dots, Y_k^d, X_k^d, Y)$$

$$= \int_{\Theta} \boxed{P_M^d(X|\theta, Y)} P_M^d(\theta|Y_1^d, X_1^d, \dots, Y_k^d, X_k^d, Y) d\theta$$

Bayes optimal classifier



e.g. linguistic analysis, hate speech detection

$$P_M^d(Y|X_1^d, Y_1^d, \dots, X_k^d, Y_k^d, X)$$

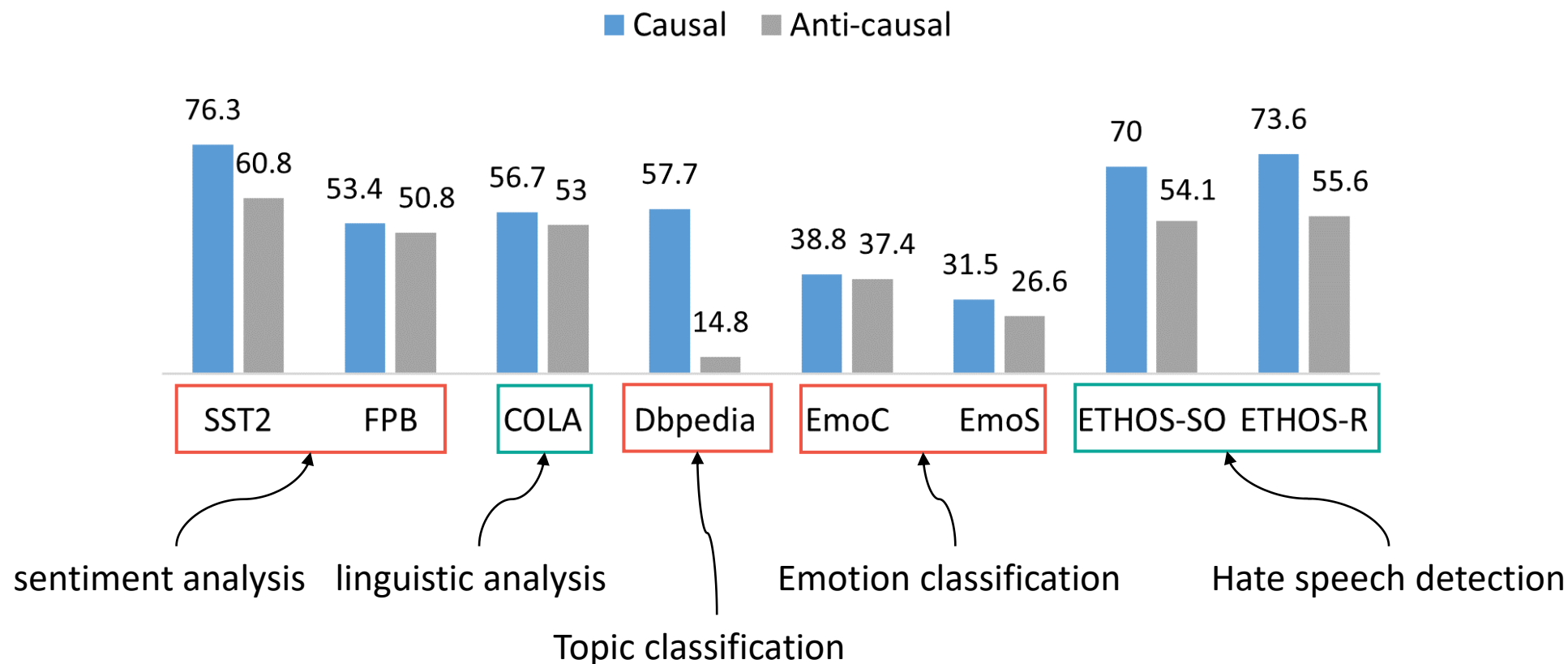
$$= \int_{\Theta} \boxed{P_M^d(Y|\theta, X)} P_M^d(\theta|X_1^d, Y_1^d, \dots, X_k^d, Y_k^d, X) d\theta$$

Bayes optimal classifier

- Assumption: the data for each task is generated by a specific value of  $\theta$ . i.e. a different value of  $\theta$  indicates a different task.



# Causal v.s. anti-causal



- 4-shot in-context learning accuracy with GPT2-large.

# Analysis in-context learning classifier

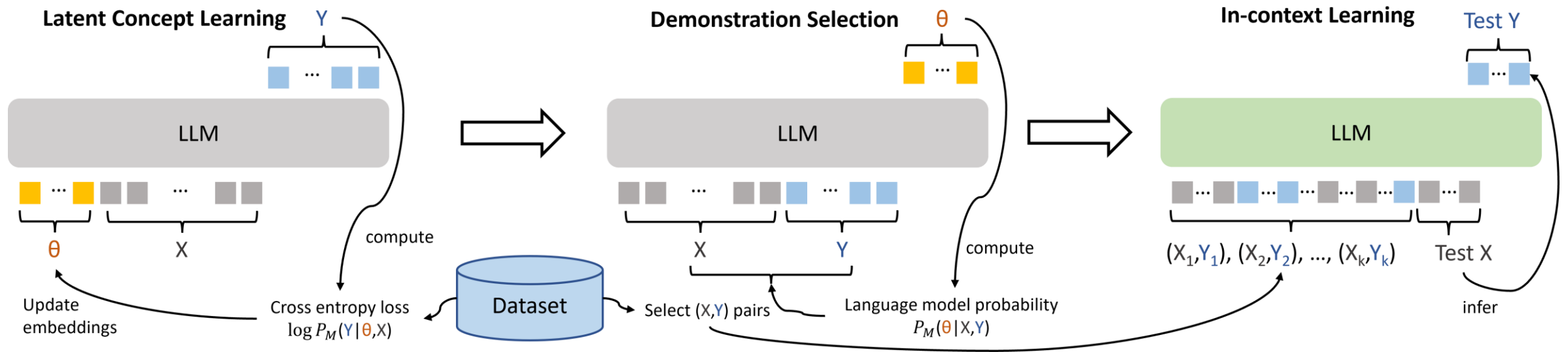
$$P_M^d(Y|X_1^d, Y_1^d, \dots, X_k^d, Y_k^d, X) \\ = \int_{\Theta} \underbrace{P_M^d(Y|\theta, X)}_{\text{Latent concept variable learning}} \underbrace{P_M^d(\theta|X_1^d, Y_1^d, \dots, X_k^d, Y_k^d, X)}_{\text{Demonstration selection}} d\theta$$

Latent concept variable learning  
(soft prompt tuning)

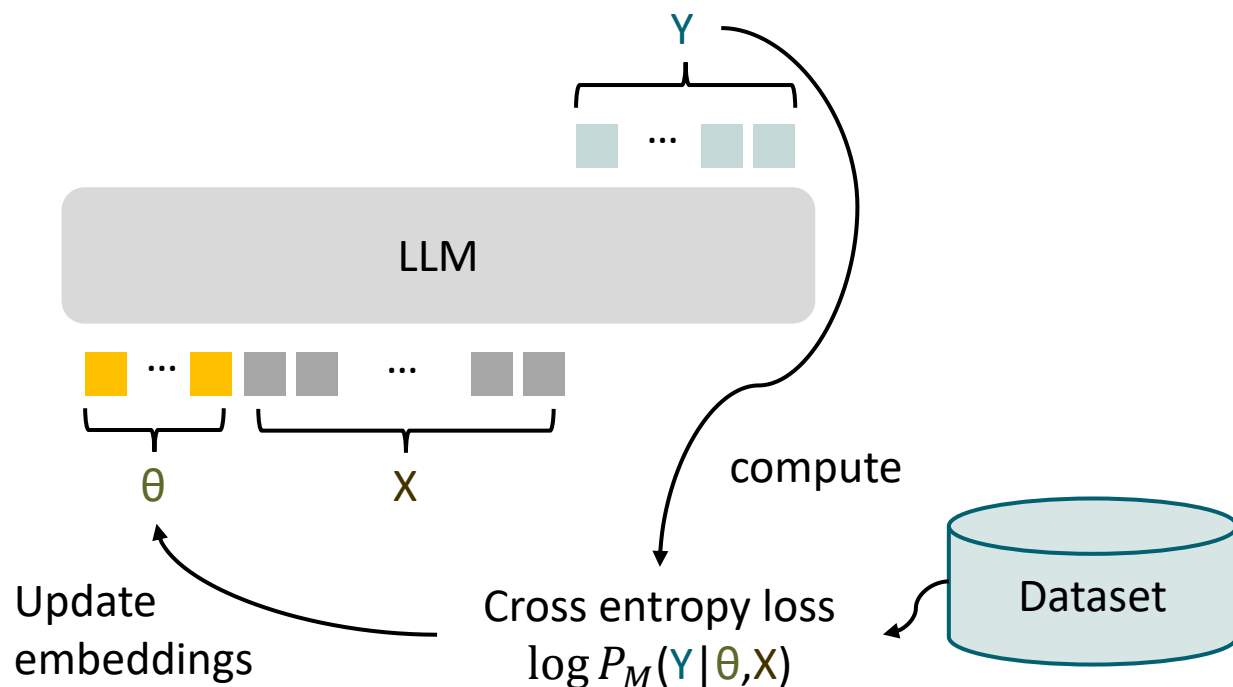
Demonstration selection

- We want to make the above in-context learning classifier  $P_M^d(Y|X_1^d, Y_1^d, \dots, X_k^d, Y_k^d, X)$  as close to the Bayes optimal classifier as possible, which means we need to make  $P_M^d(\theta|X_1^d, Y_1^d, \dots, X_k^d, Y_k^d, X)$  as concentrated on the  $\theta$  value corresponding to task  $d$  as possible.
- We can use the above conclusion to first learn a delegate of the true latent concept variable, and then use the delegate to choose the best demonstrations from a set of annotated data.

# Algorithm overview

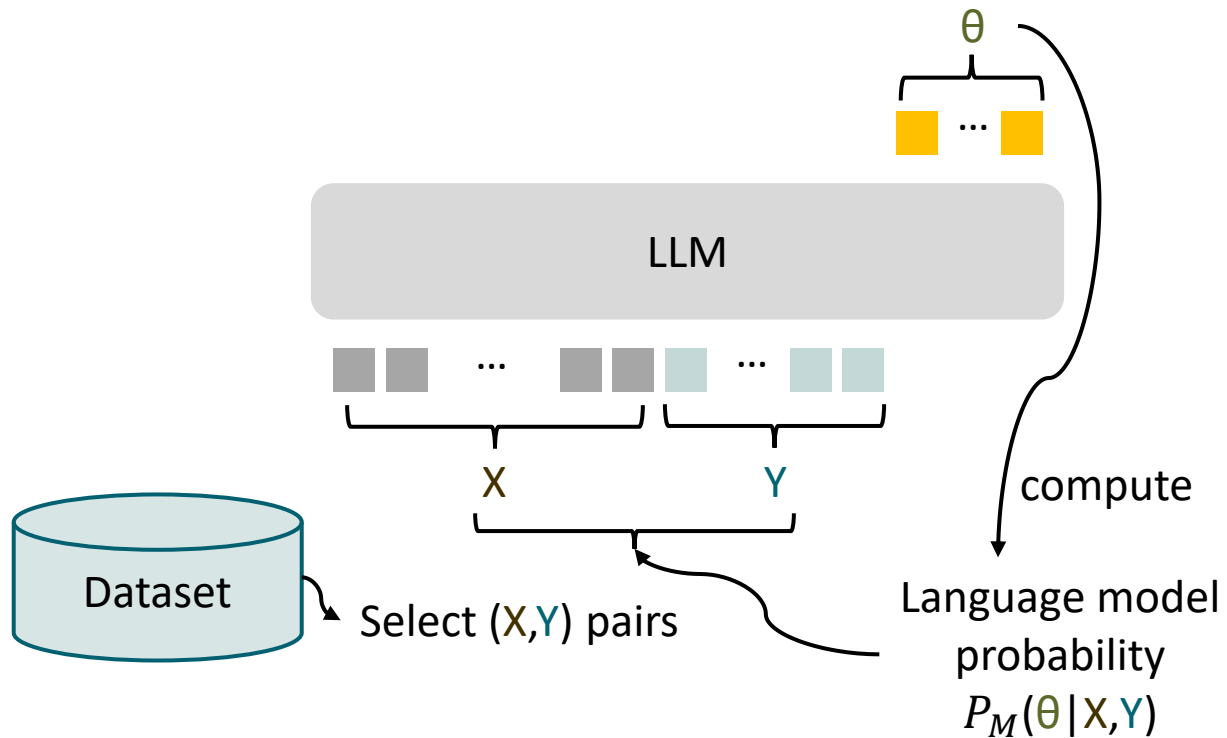


# Latent Concept Learning



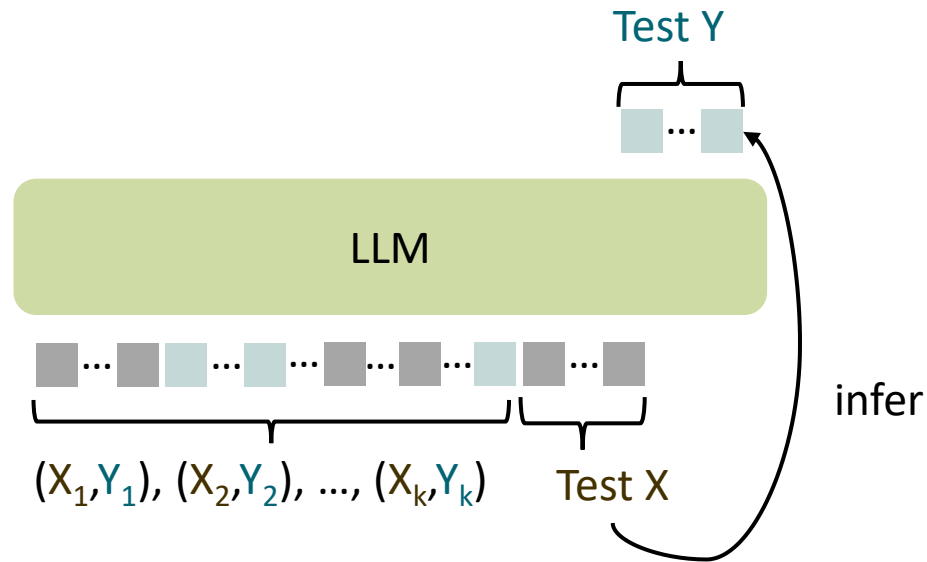
- Add a few new concept tokens to the original vocabulary of the LLM.
- Train the embedding of these concept tokens while freezing all other parameters, such that the LLM can predict the label  $Y$  given  $X$  and the concept tokens as prefixes.
- Use GPT2-large in practice.

# Demonstration Selection



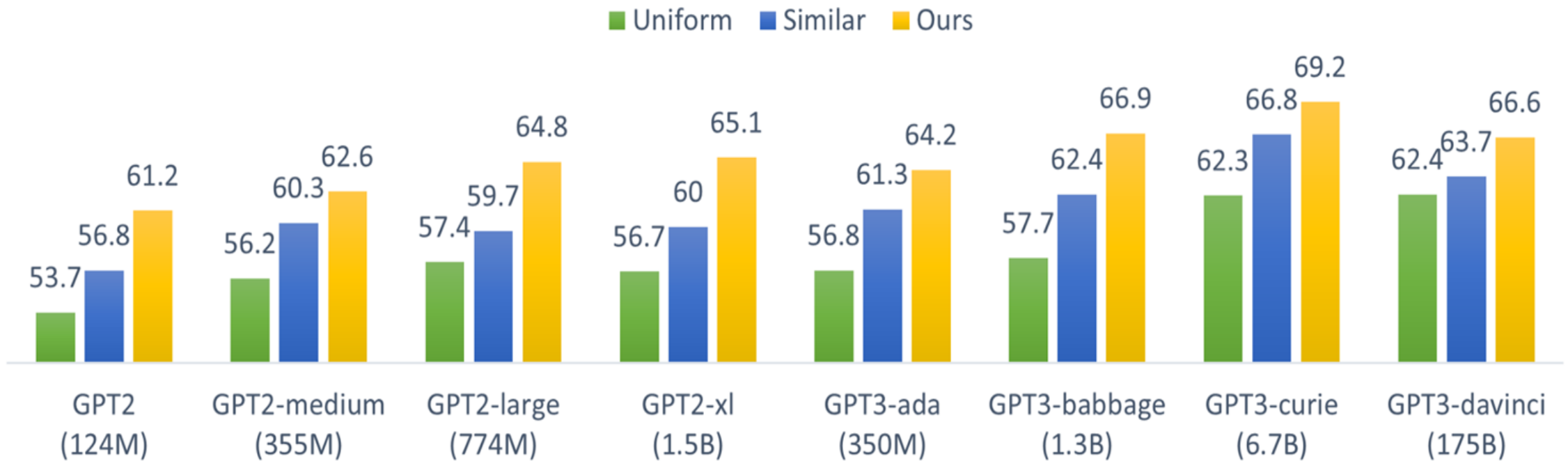
- Compute the LM probability of predicting the concept tokens given an example (X, Y).
- Then choose the top-k examples producing the highest probabilities as the demonstrations for in-context learning.
- Use GPT2-large in practice.

# In-context Learning



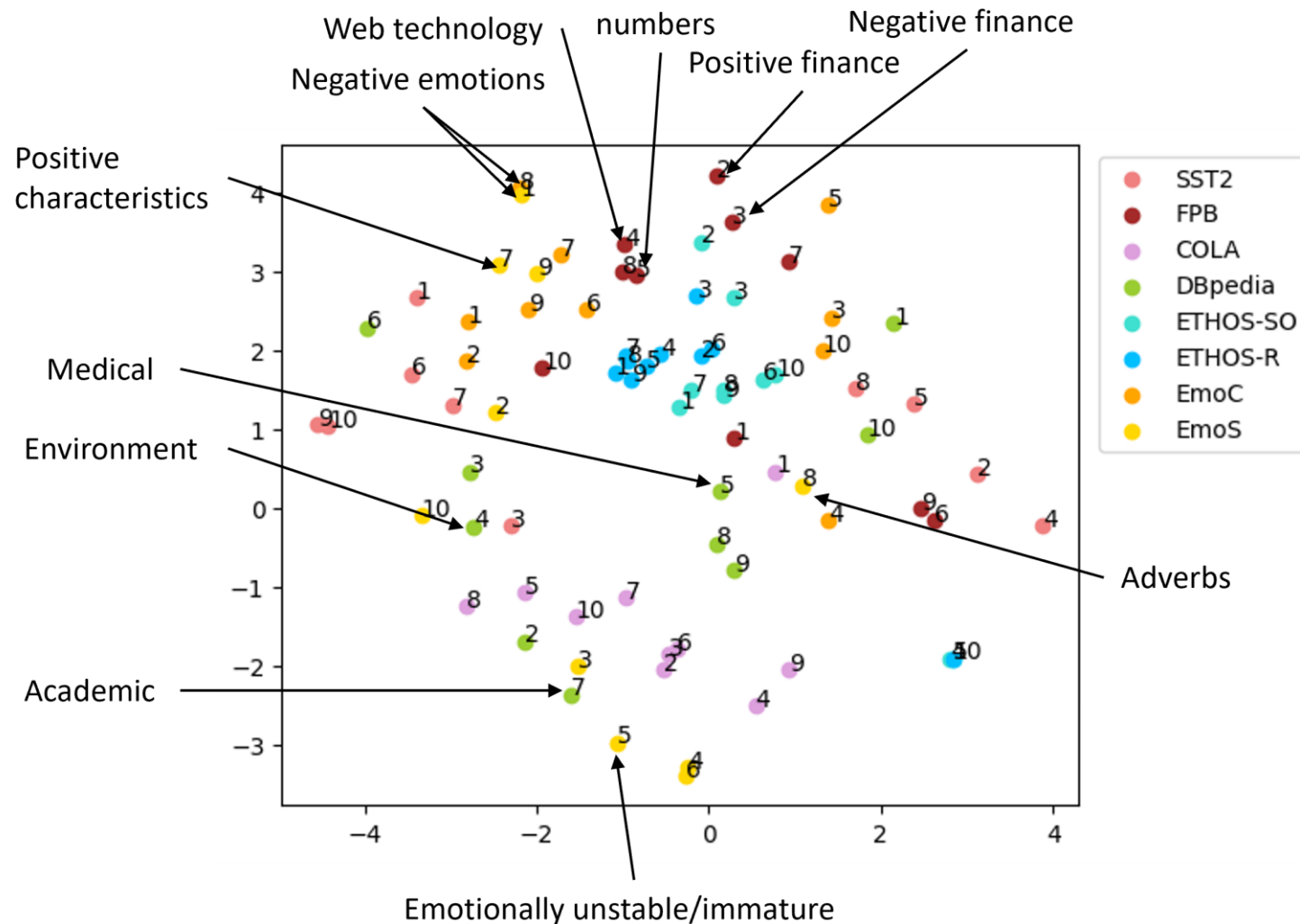
- Test the performance of the chosen  $k$  demonstrations by using them for in-context learning on a separate test set.
- Different LLMs from the previous stages can be used.
- Use different sizes GPTs in practice.

# Main results



- Results are averaged over 8 text classification datasets, each experiment is repeated by 5 runs.
- We select the optimal demonstrations by GPT2-large, and use the same set of demonstrations for all other LLMs.

# A TSNE plot of the learned concept tokens



- **SST2**: movie review sentiment analysis
- **FPB**: financial news sentiment analysis
- **COLA**: grammar error detection
- **DBpedia**: topic classification
- **ETHOS-SO** and **ETHOS-R**: hate speech detection
- **EmoC** and **EmoS**: emotion classification



# Takeaways

- Real-world LLMs implicitly infer a latent concept variable during in-context learning time.
- When have a set of annotated data, we can first learn a delegate of the concept variable and then select the demonstrations that can best represent/infer the concept variable.
- The selected demonstrations can be transferred across different-size LLMs pre-trained on similar text distributions. This indicates such behavior of LLMs comes from the pre-training data distribution.



UC SANTA BARBARA



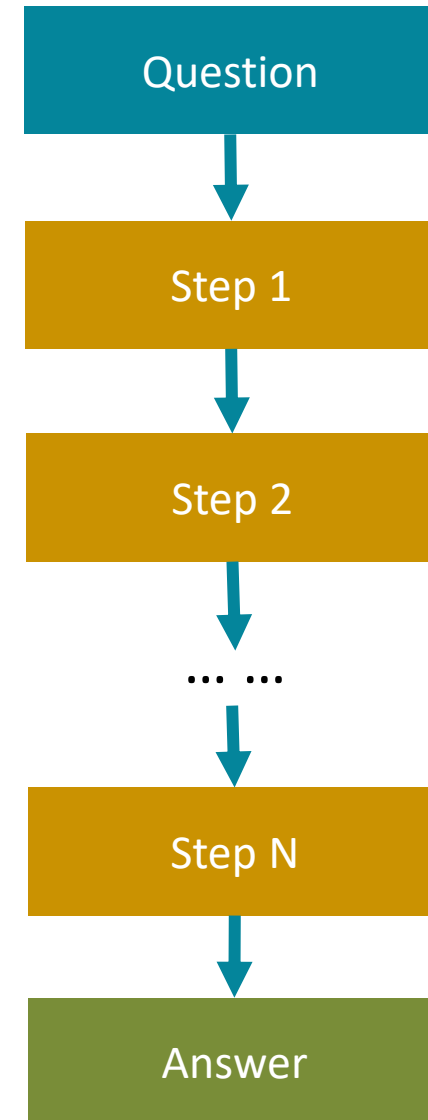
Mila

# Guiding Language Model Reasoning with Planning Tokens

Xinyi Wang, Lucas Caccia, Oleksiy Ostapenko, Xingdi Yuan, Alessandro Sordoni ([Arxiv](#))

# Chain-of-thoughts reasoning

- Question: Every day, Wendi feeds each of her chickens three cups of mixed chicken feed, containing seeds, mealworms and vegetables to help keep them healthy. She gives the chickens their feed in three separate meals. In the morning, she gives her flock of chickens 15 cups of feed. In the afternoon, she gives her chickens another 25 cups of feed. How many cups of feed does she need to give her chickens in the final meal of the day if the size of Wendi's flock is 20 chickens?
- If each chicken eats 3 cups of feed per day, then for 20 chickens they would need  $3 \times 20 = 60$  cups of feed per day.
- If she feeds the flock 15 cups of feed in the morning, and 25 cups in the afternoon, then the final meal would require  $60 - 15 - 25 = 20$  cups of chicken feed.
- The answer is: 20



# Planning errors made by LLMs

Question: Every day, Wendi feeds each of her chickens three cups of mixed chicken feed, containing seeds, mealworms and vegetables to help keep them healthy. She gives the chickens their feed in three separate meals. In the morning, she gives her flock of chickens 15 cups of feed. In the afternoon, she gives her chickens another 25 cups of feed. **How many cups of feed does she need to give her chickens in the final meal of the day** if the size of Wendi's flock is 20 chickens?

## Vanilla supervised fine tuning

Every day, Wendi gives her chickens 15 cups of feed in the morning + 25 cups of feed in the afternoon =  $\langle\langle 15+25=40 \rangle\rangle 40$  cups of feed.

If she has 20 chickens and she feeds them 40 cups of feed every day, then each chicken gets  $40/20 = \langle\langle 40/20=2 \rangle\rangle 2$  cups of feed per chicken.

The answer is: 2

*\* Drifting away from the correct reasoning flow*

## Fine tuning with arithmetic planning tokens

$\langle\text{prefix}\rangle \langle+\rangle$  Wendi gives her flock 15 cups of feed in the morning and another 25 cups in the afternoon, for a total of  $15+25 = \langle\langle 15+25=40 \rangle\rangle 40$  cups of feed.

$\langle\text{prefix}\rangle \langle*\rangle$  If Wendi has 20 chickens, then she needs  $20*3 = \langle\langle 20*3=60 \rangle\rangle 60$  cups of feed to feed her flock.

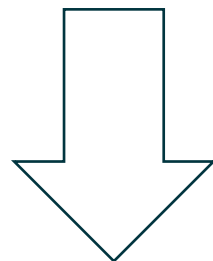
$\langle\text{prefix}\rangle \langle-\rangle$  If Wendi has already given her flock 40 cups of feed, then she needs to give her flock  $60-40 = \langle\langle 60-40=20 \rangle\rangle 20$  more cups of feed.

$\langle\text{prefix}\rangle \langle\text{answer}\rangle$  The answer is: 20

# A Bayesian view of chain-of-thoughts

Latent variable distribution only drastically changes at the beginning of each CoT step

$$P_M(\mathbf{w}_{t+1:T}|\mathbf{w}_{1:t}) = \int_{\Theta} P_M(\mathbf{w}_{t+1:T}|\boldsymbol{\theta}) P_M(\boldsymbol{\theta}|\mathbf{w}_{1:t}) d\boldsymbol{\theta}$$



**Simplified assumption:** there is a discrete planning variable governing each chain-of-thoughts step.

# Planning tokens

General  
planning  
tokens

<prefix>

<+>

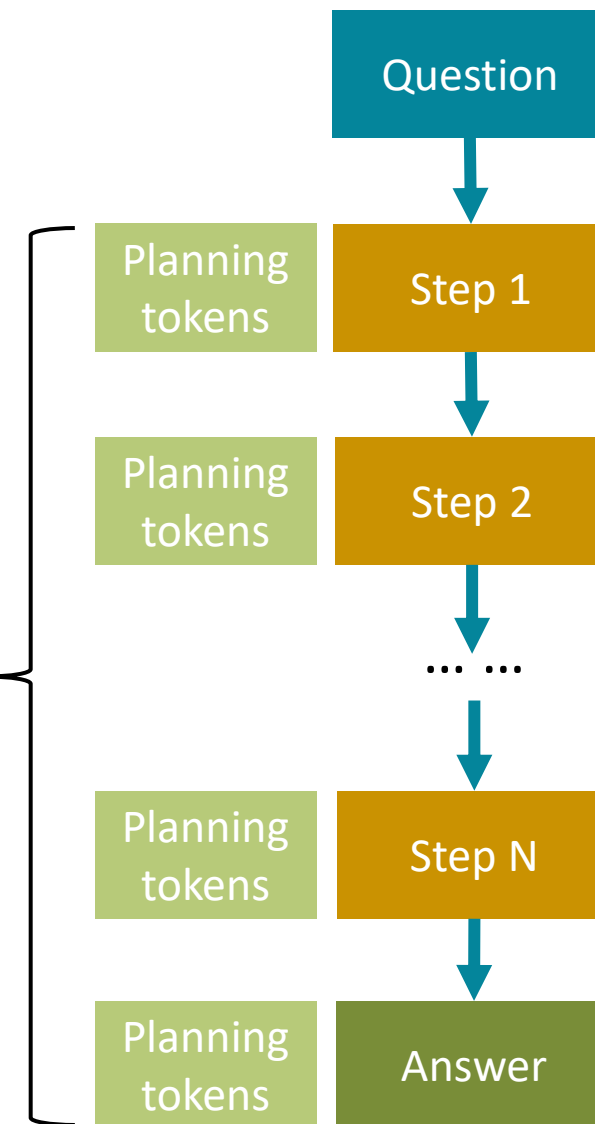
Specialized planning tokens  
Wendi gives her flock 15 cups of feed in the morning and another 25 cups in the afternoon, for a total of  $15+25 = <<15+25=40>>40$  cups of feed.

<prefix> <\*> If Wendi has 20 chickens, then she needs  $20*3 = <<20*3=60>>60$  cups of feed to feed her flock.

<prefix> <-> If Wendi has already given her flock 40 cups of feed, then she needs to give her flock  $60-40 = <<60-40=20>>20$  more cups of feed.

<prefix> <answer> The answer is: 20

Idea:  
better  
control of  
reasoning  
flow



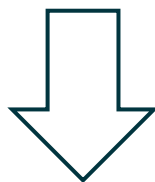
# Training with planning tokens

**Step classifier**  
(light weight)

$$P_{\phi}(T_i|S_i, S_{context})$$

$$T_i \in \Omega$$

Assign a planning token to each reasoning step in the training set



Plan

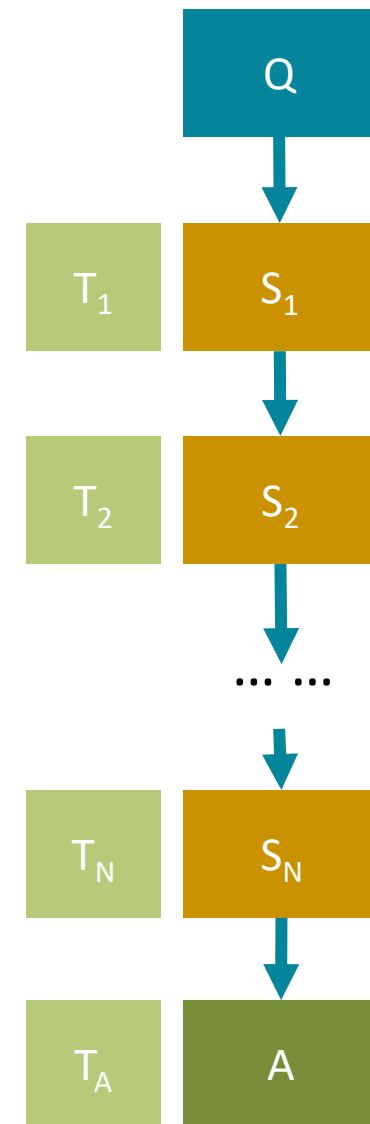
Generate

**LLM**  
(very large)

$$\prod_{i=1}^N [P_{\theta}(T_i|Q, T_1, S_1, \dots, T_{i-1}, S_{i-1}) P_{\theta}(S_i|Q, T_1, S_1, \dots, T_{i-1}, S_{i-1}, T_i)]$$

Only tune the embeddings of the planning tokens and some helping adapters (e.g. LORA)

\* Note: to increase the representation capacity, the planning token can be a sequence of tokens.



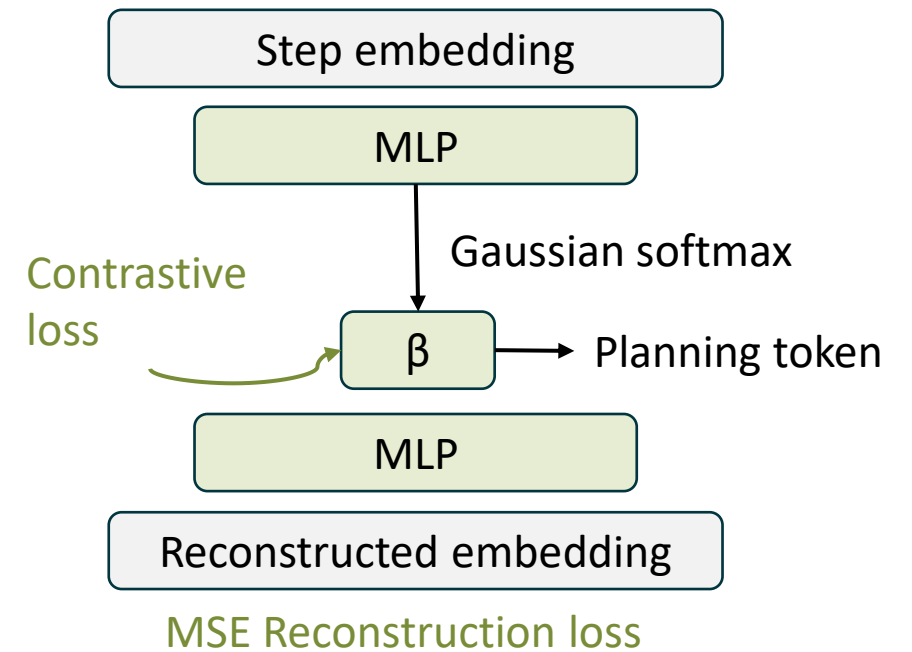
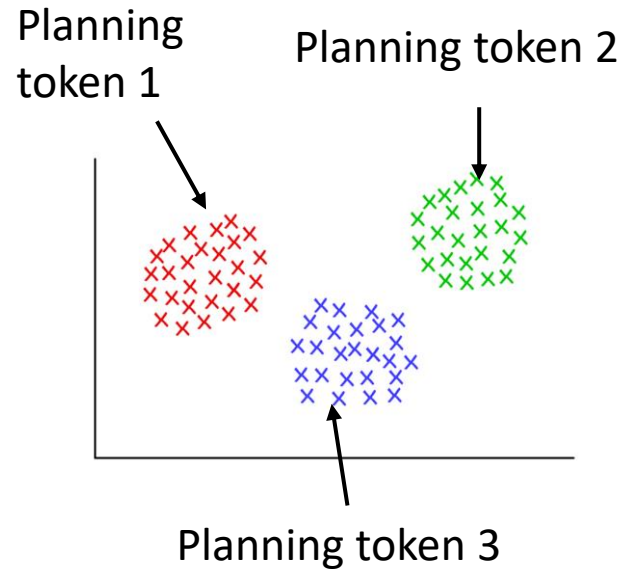
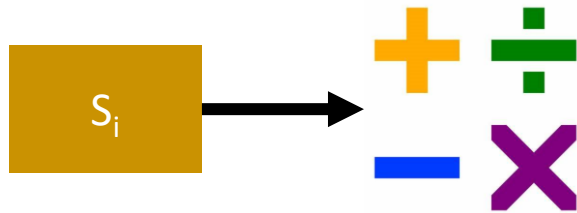
# Step classifiers

\* Reasoning steps embedded with T5 encoder

Arithmetic

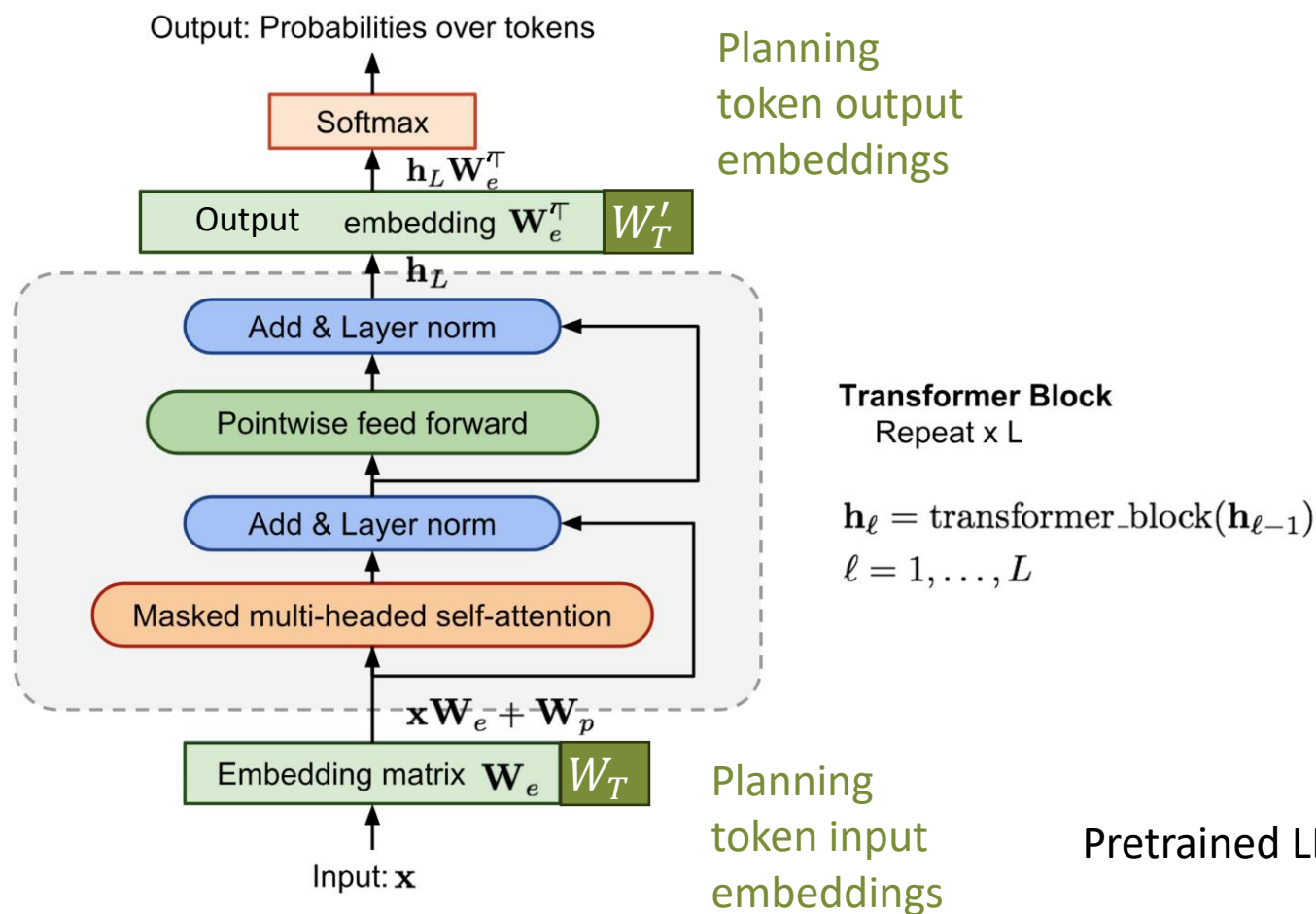
Clustering

Latent



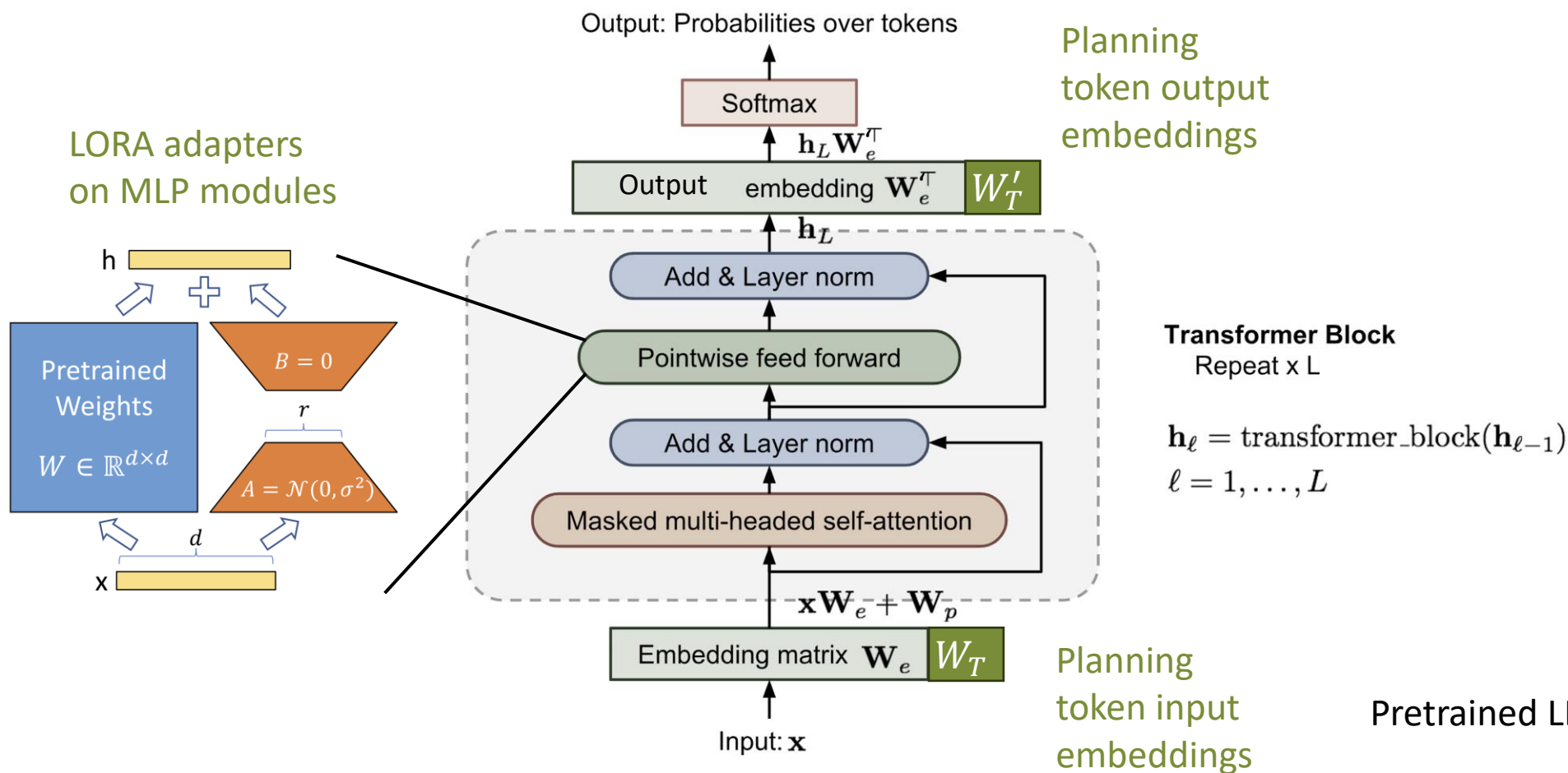


# Planning tokens implementation



Pretrained LLM: Llama 2 (7B)

# Combination with parameter efficient tuning



# Main results

Base model	Planning Type	#clusters	#trainable	GSM8K	MATH	AQUA	Avg
Phi 1.5 (1.3B)	N/A	0	100%	12.5	1.3	27.2	13.5
	General	1	100%	15.4	2.0	35.4	17.6
	Arithmetic	4	100%	15.0	2.3	33.1	16.8
	K-Means	5	100%	14.5	2.7	<b>36.5</b>	17.7
	SQ-VAE	5	100%	<b>15.8</b>	<b>3.3</b>	34.3	<b>17.8</b>
Llama2 (7B)	N/A	0	0.343%	38.2	6.5	36.6	27.1
	General	1	0.344%	38.5	6.7	37.8	27.7
	Arithmetic	4	0.344%	39.5	5.6	38.2	27.8
	K-Means	5	0.344%	39.1	6.7	40.5	28.8
	SQ-VAE	5	0.344%	<b>40.0</b>	<b>7.0</b>	<b>41.3</b>	<b>29.4</b>
Llama2 (13B)	N/A	0	0.279%	44.6	7.2	41.3	31.0
	General	1	0.280%	47.9	7.9	42.5	32.8
	Arithmetic	4	0.280%	41.9	4.6	35.8	27.4
	K-Means	5	0.280%	49.6	8.4	<b>44.1</b>	34.0
	SQ-VAE	5	0.280%	<b>50.6</b>	<b>8.5</b>	43.9	<b>34.3</b>

Table 1: Testing accuracy of fine-tuned language models on different math word datasets.

# Reasoning length effect

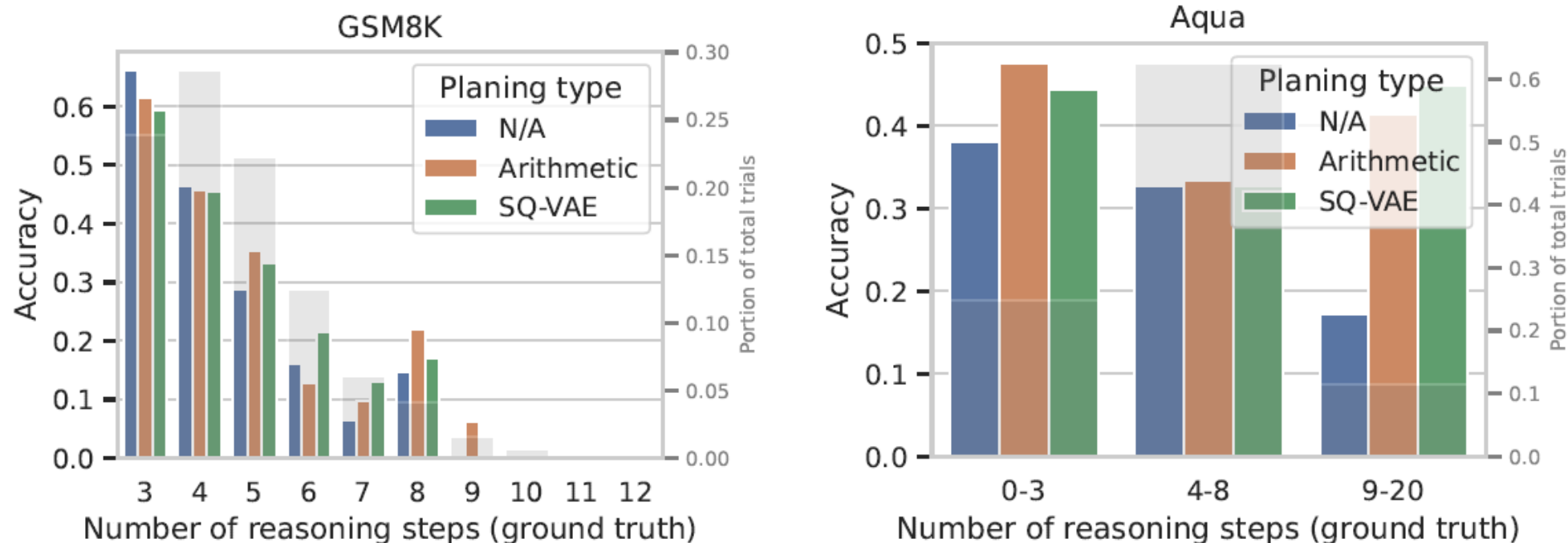


Figure 2: Accuracy on GSM8K (**left**) and Aqua (**right**) on test examples by their number of ground-truth reasoning steps. SQ-VAE consistently increases performance for test examples that require more steps of reasoning to be solved.

# Error types predicted by GPT4

Error type	Misunderstanding of question	Computation errors	Inaccurate extraction of question information	Wrong application of math knowledge	Wrong logic
Example	Question: ...How many cups of feed does she need to give her chickens in the final meal of the day... ...then each chicken gets 40/20 = <<40/20=2>>2 cups of feed per chicken. The answer is: 2	...7 + 3301x = 3371x...	Question: ...He spends the next half-hour driving at a speed of 30mph... ...He drove 2 hours at 30mph so he traveled 2*30=<<2*30=60>>60 miles...	...The number of feet the plane is from the ground is the target of a geometric sequence...	...\$2ab = 12 = 2^3 \cdot b^2\$...
N/A	17	48	55	4	48
SQ-VAE	21	44	42	4	50

# Takeaways

- Planning tokens improve LLM's math reasoning performance by (a) increasing the reasoning sequence capacity, (b) making LLM a mix-of-expert model for different reasoning types.
- Planning tokens improve the intra-step consistency.
- Concurrent work shows that similar method (our general baseline) also works on QA tasks at pre-training time ([Goyal et al. 2023](#)).

**Thank you!**

Questions?