Understanding Pre-trained Large Language Models through a Probabilistic Lens

Xinyi Wang (Major Area Exam)
Committee: William Wang, Kun Zhang, Shiyu Chang, Xifeng Yan

Outline

- Background on large language models
- Recent works on understanding large language models
- Future directions and current progress
- Q&A

Background on large language models

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

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

N-gram model

$$p(w_1, w_2, \dots, w_T) = \prod_{i=1}^{T} p(w_i | w_{i-1}, w_{i-2}, \dots, w_{i-N})$$

Hidden Markov model

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

Language Model

Classic language models:

Counting

Bag of words model

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

N-gram model

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

Hidden Markov model

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

Dynamic programming with fixed transition matrix

Neural language models:

Word embedding model

$$p(w_1, w_2, \dots, w_T)^{2c} = \prod_{i=1}^{T} \prod_{-c \le j \le c, j \ne 0} p(w_{i+j} | w_i)$$

Effectively an embedding layer followed by one-layer fully-connected neural network with softmax activation

RNN, LSTM, Transformer (w/. decoder)

Transformer (w/. encoder)

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

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

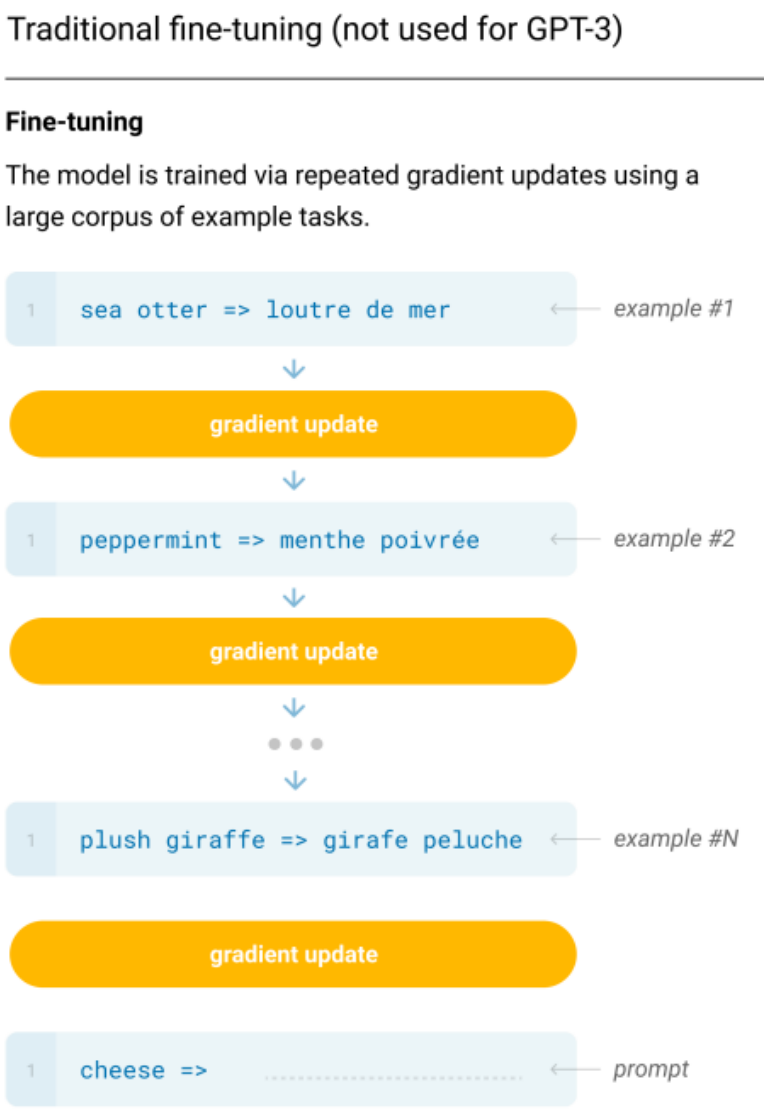
Beyond probability estimation

- While language models are trained to estimate the previous text sequence distribution, the interesting part is that they are shown to be useful beyond distribution modeling.
- **Word2Vec (Mikolov et al., 2013)**: a non-contextual word embedding model, using a simple fully-connect neural network.
  - Serves as a significantly better feature for many NLP tasks. Achieves State-of-the-art (SOTA) performance (at that time) on many NLP tasks.
  - It appears that the analogy between words can be expressed as simple arithmetic in the Word2Vec embedding space. E.g. King – Man = Queen - Woman

Large language model

- **BERT (Devlin et al., 2018)**: a pre-trained masked language model, using an encoder-only Transformer architecture.
  - Serves as a good initialization for many downstream NLP tasks.
  - SOTA performance (at that time) on many NLP tasks can be achieved by fine-tuning BERT on corresponding training sets.
- **GPT3 (Brown et al., 2020)**: a pre-trained generative language model, using a decoder-only Transformer architecture.
  - Serves as a general NLP task solver itself.
  - SOTA or close to SOTA performance (at that time) on many NLP tasks can be achieved by few-shot, even zero-shot prompting at inference time *without any parameter updating*.

Fine-tuning



- Use the pre-trained large language model as a good starting point for learning downstream NLP tasks.
- Expensive to train when the model is large.
- Training data required (not necessarily a large amount).
- Parameter efficient fine-tuning: only tune a small number of parameters in the model and fix other parameters.
  - Soft prompt tuning (Lester et al., 2021): add a few trainable new tokens at the beginning of each sequence for a specific task and fix all other parameters.
  - Head tuning (Peters etal., 2018): learning a classifier on top of the frozen pre-trained model.
  - Usually match the performance of full fine-tuning with significantly less computation.

In-context learning

The three settings we explore for in-context learning

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
```

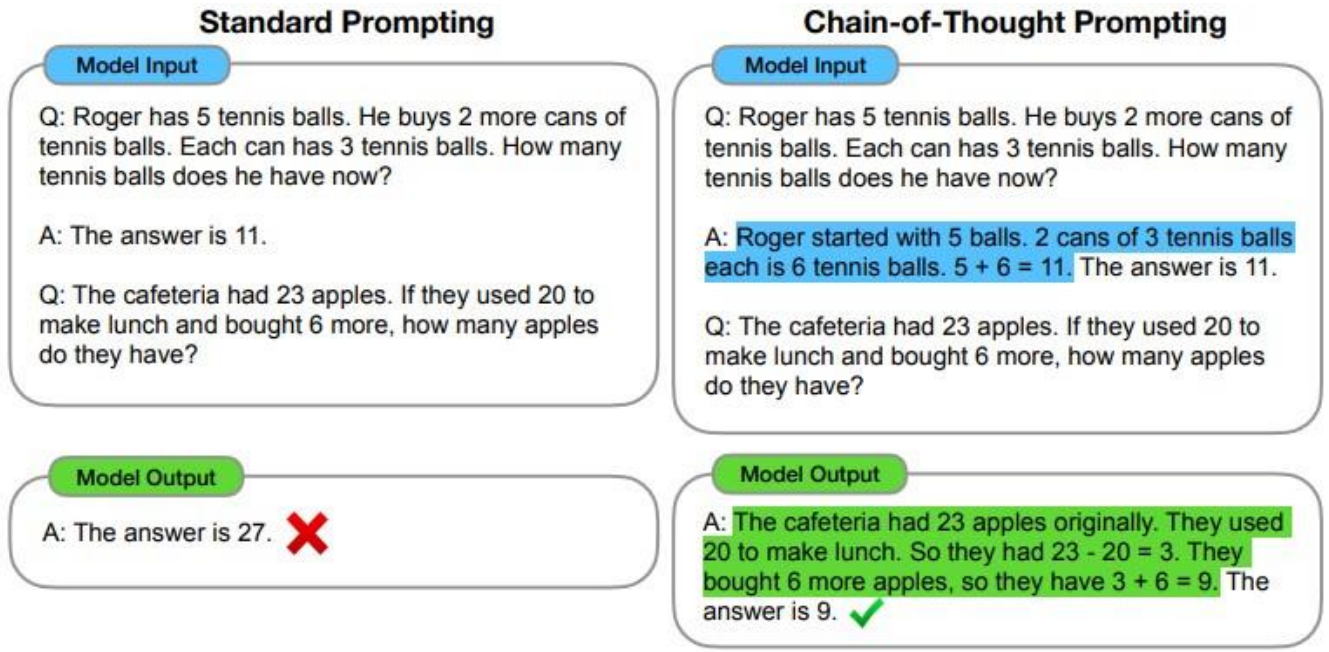
Few-shot

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
```

(Brown et al., 2020)

- Only works well for large enough generative language models (e.g. 175B GPT3).
- Most common way to interact with pre-trained large language models nowadays.
- Can be combined with chain-of-thoughts prompting (Wei et al., 2022).



(Wei et al., 2020)

Exponential scaling law

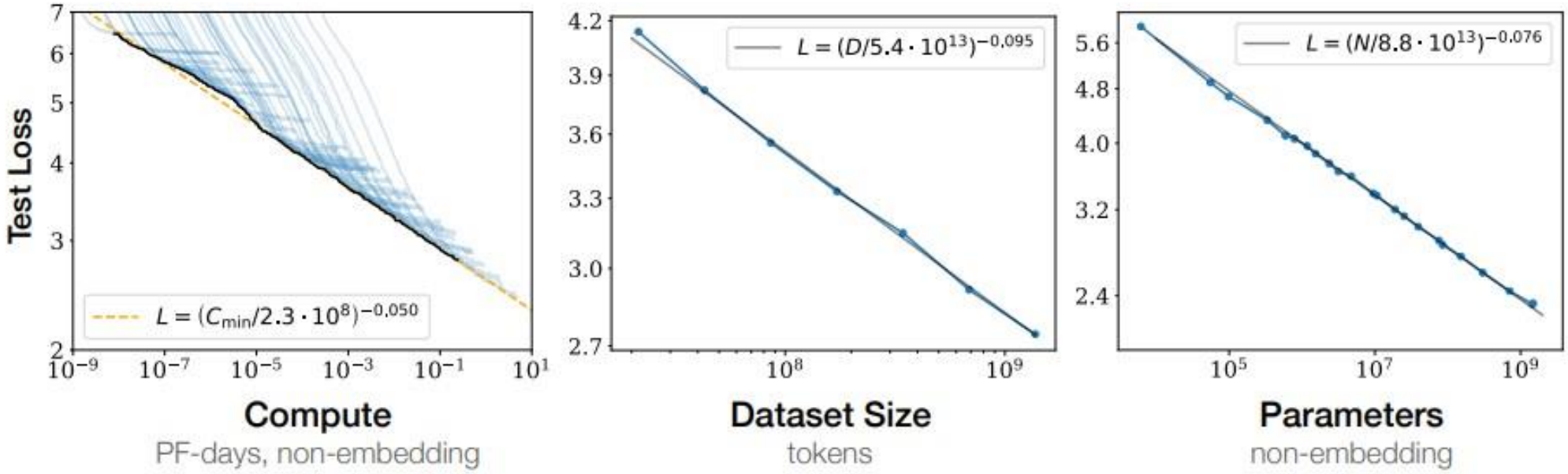
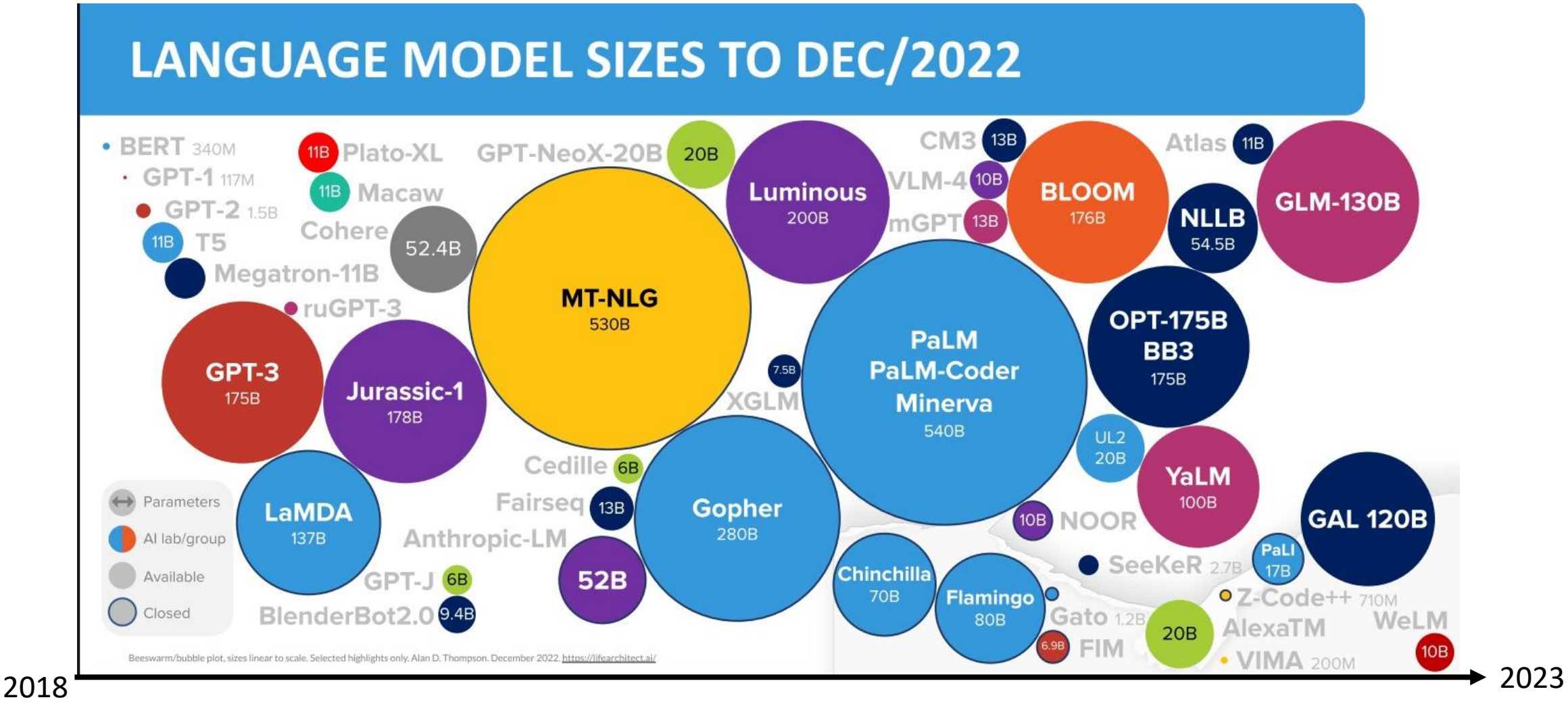


Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

- Experiments performed using GPT-like models: decoder-only Transformer, generative language modeling objective. (Kaplan et al., 2020)
- The language model performance is measured by cross-entropy loss over a test set.

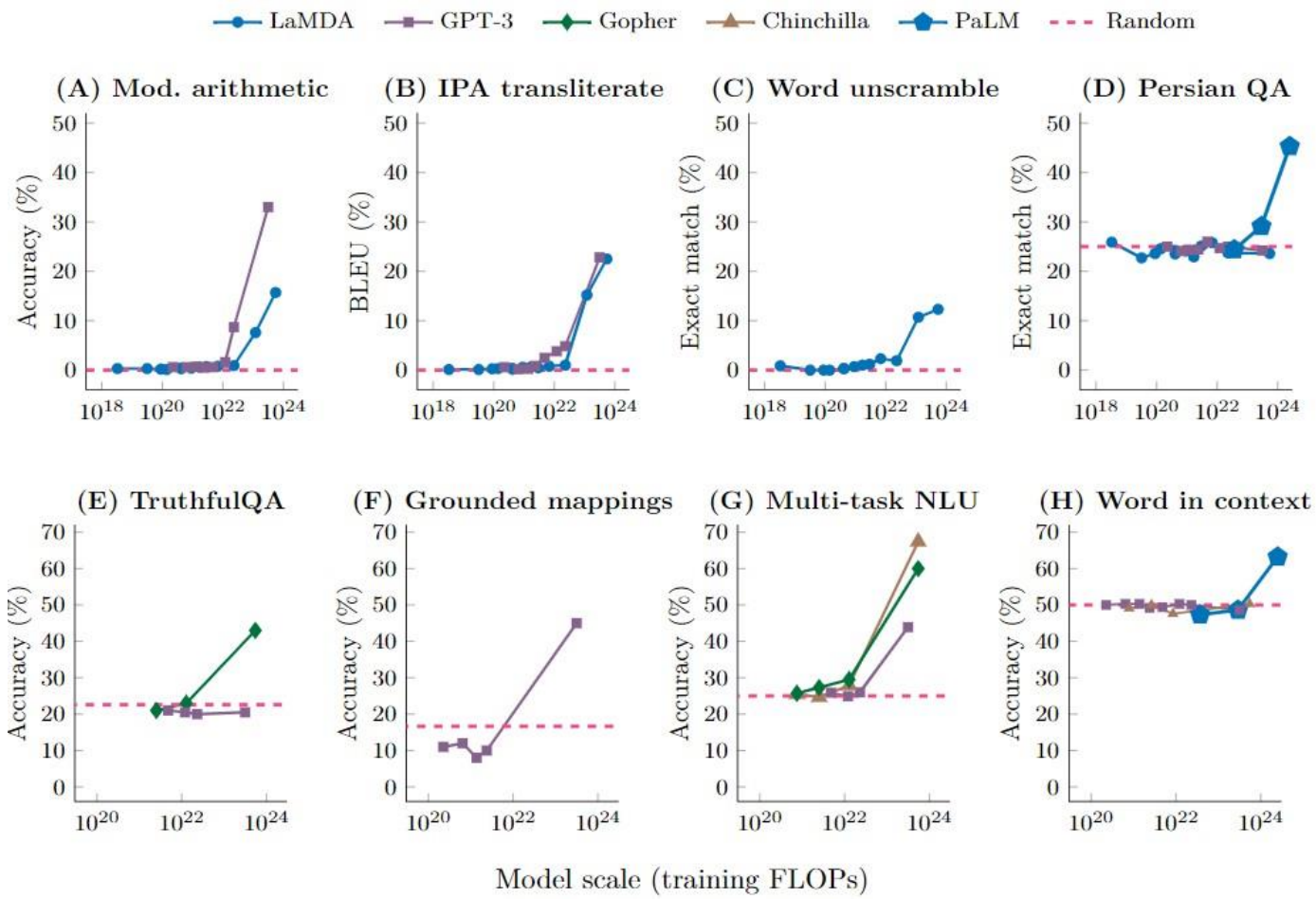
Existing large language models



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

Emergent abilities

- **Definition**: An ability is emergent if it is not present in smaller models but is present in larger models. (Wei et al., 2022)
- The performance is near-random until a certain critical threshold of scale is reached, after which performance increases to substantially above random.
- Examples:
  - Few-shot prompting (in-context learning) for arithmetic, truthful QA, etc.
  - Chain of thought prompting for solving math word problems.
  - Instruction following with instruction fine-tuning.



Recent works on understanding large language models

How to understand these phenomena?

- Large language models (LLMs) are black-box deep neural networks that are hard to know their mechanism inside.
- The best-performing LLMs are either not open source (e.g. PaLM) or only their APIs are released (e.g. GPT3).
- *Fine-tuning*: detailed later.
- *Prompting/in-context learning*: detailed later.
- *Exponential scaling law:*
  - A general empirical law for deep neural networks (Hestness, et al., 2017; Rosenfeld et al., 2020).
  - Theoretically, the power-law generalization error rate is well-known for linear/kernel models (Caponnetto and De Vito, 2007).
  - There are some theoretical works towards this direction, though usually for fully-connect neural networks (Schmidt-Hieber, 2020; Suzuki, 2018).
- *Emergent abilities*: no good explanations.

Understanding fine-tuning

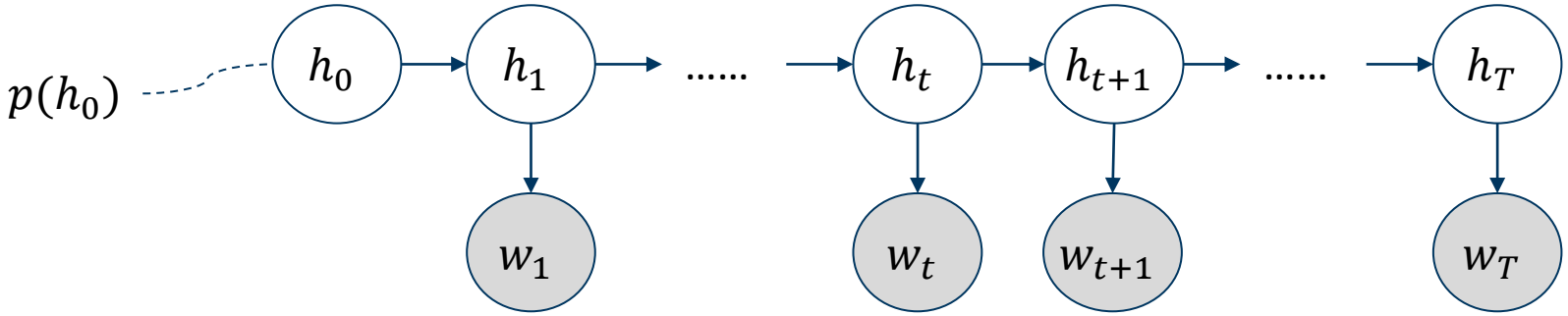
- Natural task: the distribution of the next word, conditional on the context, can provide a strong discriminative signal for the downstream task (Saunshi et al., 2021).
  - Assumption: downstream labels are recoverable via a linear head applied to the conditional token probabilities.
  - Experiments: use data from a simple task. E.g. linear regression.
- **Hidden Markov Model data distribution**: the first hidden state contains all the required information to recover downstream task labels.
  - *Why Do Pretrained Language Models Help in DownstreamTasks? An Analysis of Head and Prompt Tuning*. Wei et al. NeurIPS 2021.
  - Experiments: use data generated from a fixed distribution.

Understanding in-context learning

- Examine pre-training data distribution:
  - **Hidden Markov Model**
    - *An Explanation of In-context Learning as Implicit Bayesian Inference* by Xie, et al. Appears in ICLR, 2022.
  - Unique distributional properties of language (Chan et al., 2022)
    - long tail, the dynamic meaning of words, etc
  - Experiments: use data generated from a fixed distribution.
- Mechanism of utilizing the few-shot demonstrations:
  - Mimic gradient descent at inference time (von Oswald et al. 2022)
  - Smaller models are encoded in activation (Akyurek et al. 2022)
  - Transformer itself is a learning algorithm (Li et al., 2023)
  - Experiments: use data from a simple task. E.g. linear regression.

Hidden Markov Model data distribution

Assuming pre-training data distribution p :
(Wei et al., 2021; Xie et al., 2022)



- Let word tokens $w_i \in V$, where V is the vocabulary.
- Let hidden states $h_i \in H$.
- The transition probabilities $p(h_i|h_{i-1})$ and the emission probabilities $p(w_i|h_i)$ are time-invariant.
- The joint probability of token sequence $w_{1:T}$ and latent states $h_{0:T}$ can be written as: $p(w_{1:T}, h_{0:T}) = p(h_0) \prod_{i=1}^{T} p(w_i|h_i)p(h_i|h_{i-1})$

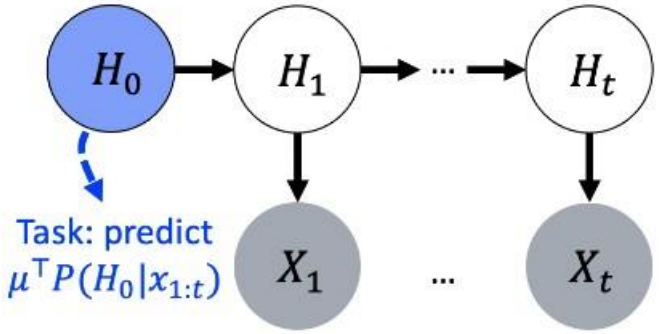
Head tuning and prompt tuning

- Consider a BERT-like **masked language model** M.
- Let the first token w_1 be the special token [CLS]. Let $x = w_{2:T}$ denote the input text sequence and y denote the discrete output label. $F^*(x) = y$.
- Head tuning means learning a linear classifier on top of the output of the first [CLS] token, $p(w_1|w_{2:T})$.
- **Want**: head tuning can learn the ground truth classifier F^*.
- Assumptions: (Wei et al., 2021)
  - The language model M perfectly learns the text distribution $p(w_i|w_1, \dots, w_{i-1}, w_{i+1}, \dots, w_T)$.
  - h_0 has all the meaningful information for the downstream task.
    - i.e. Ground truth mapping $y = F^*(x)$ is assumed to be a linear classifier on the posterior $p(h_0|x)$.
  - The Markov chain $h_{0:T}$ is ergodic, and $p(h_0) > 0$ for all $h_0 \in H$.
  - The token emission probability matrix $P(W_i|H_i)$ has linearly independent columns. i.e. $|H| \leq |V|$.

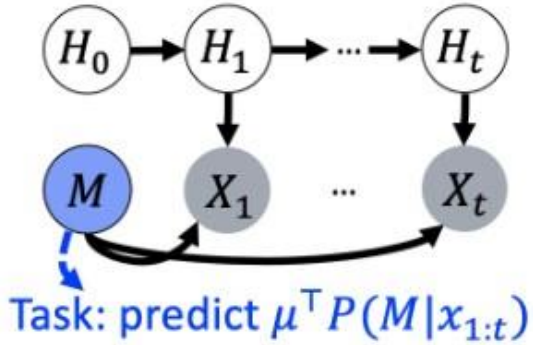
Head tuning and prompt tuning

- Since $p(w_1|w_{2:T}) = \sum_{h_1 \in H} p(w_1|h_1)p(h_1|w_{2:T})$ can be written in a matrix form $P(W_1|W_{2:T}) = P(W_1|H_1)P(H_1|W_{2:T})$. Since $P(W_i|H_i)$ is invertible, $P(H_1|W_{2:T})$ can be written as a linear transformation of $P(W_1|W_{2:T})$.
- Since the first [CLS] token itself does not have semantic meaning, $P(H_1|W_{2:T})$ can be expressed as a linear function of $P(H_0|W_{1:T})$. Thus, the ground truth classifier $F^*(x)$ can be expressed as a linear classifier on $p(w_1|w_{2:T})$.
- If we also use soft prompt tuning, i.e. prepend a trainable new token to the text sequence, then the last assumption on the invertible token emission probability matrix can be relaxed. i.e. the total number of hidden states can be larger than the vocabulary size, which increase the modeling capacity of the HMM.

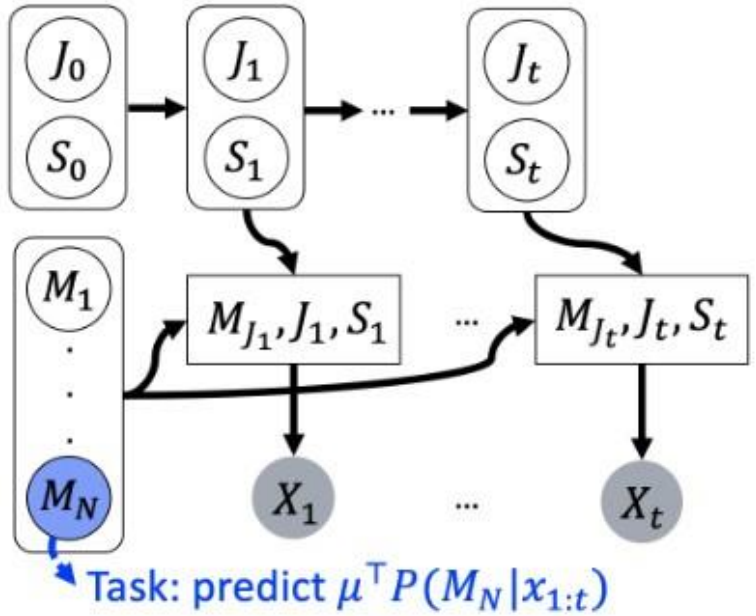
Head tuning and prompt tuning



HMM



Memory-augmented HMM
with a single memory cell



Memory-augmented HMM
with multiple memory cells

- More realistic scenario with long-term dependency can be modeled with memory augmented HMMs.
- With relaxed assumptions, the ground truth classifier $F^*(x)$ can be expressed as an attention over the whole output sequence.

Head tuning and prompt tuning

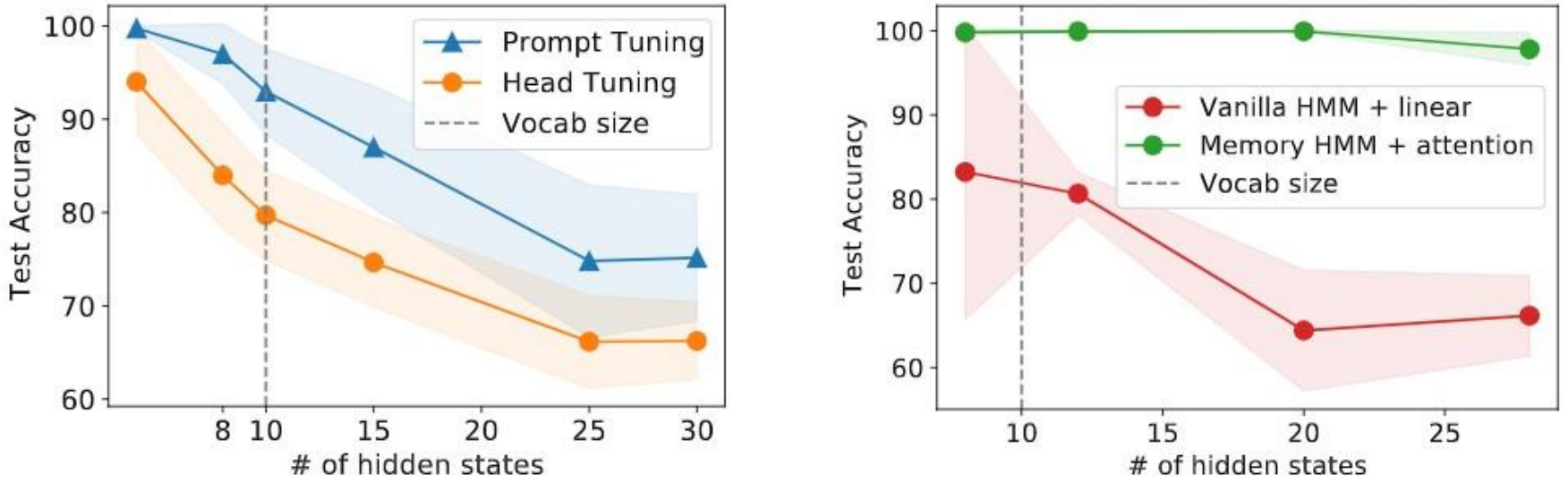
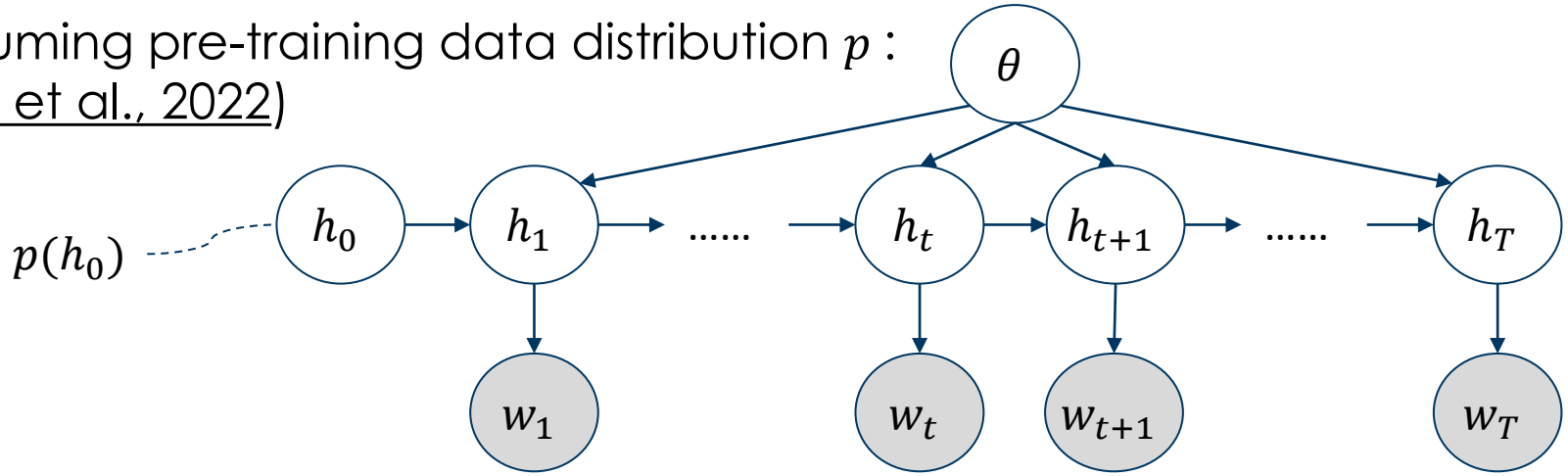


Figure 3: **Left:** Head vs. prompt tuning with a linear head on synthetically-generated HMM data, with varying hidden state sizes. Prompt tuning improves downstream accuracy especially when the problem is degenerate ($|\mathcal{H}| > |\mathcal{X}|$). **Right:** Downstream accuracy of head tuning on data from vanilla HMM vs. memory-augmented HMM, across varying values of $|\mathcal{M}||\mathcal{H}|$. Long-term dependencies in the memory-augmented HMM data improve downstream recovery when using attention. Experiments average over 20 trials (left) and 5 trials (right) of pretraining and finetuning, with 95% intervals shown.

- Experiments on synthetic data using a BERT-like language model.

In-context learning

Assuming pre-training data distribution p :
(Xie et al., 2022)

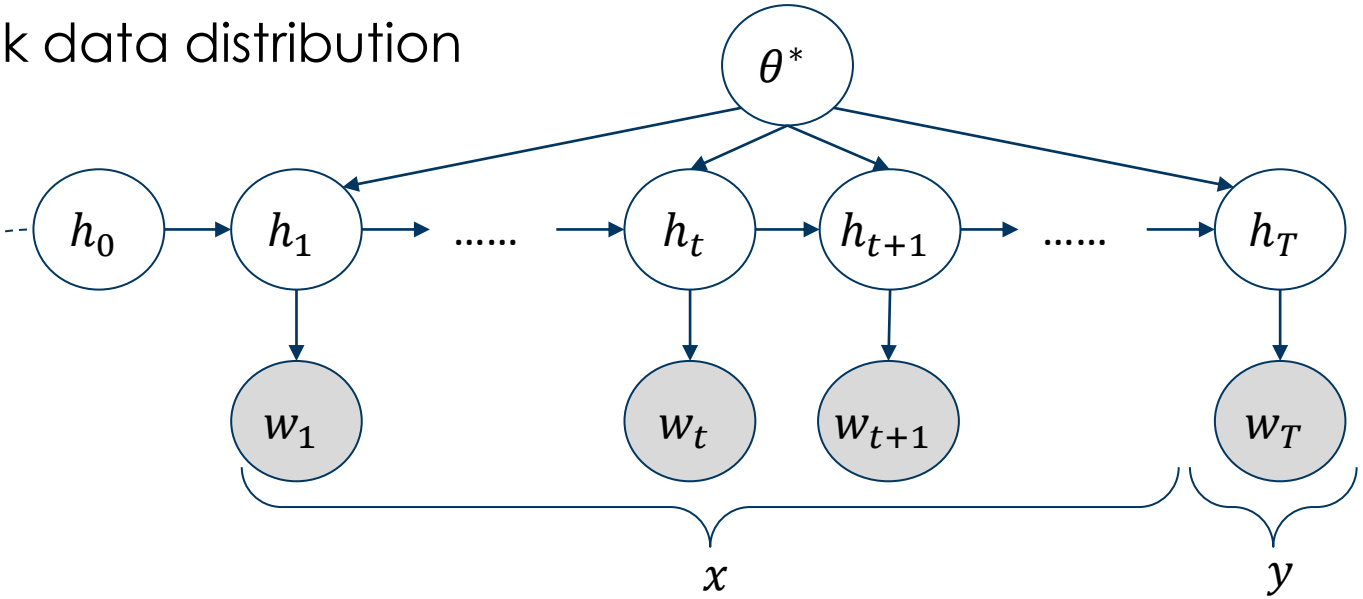


- Consider a GPT-like **generative language model** M.
- Let $\theta \in \Theta$ denote a latent concept variable.
- Assumptions:
  - The language model M perfectly learns the text distribution $p(w_i|w_1, \ldots, w_{i-1})$.
  - θ determines the transition probability.
- The probability of token sequence $w_1, w_2, \ldots, w_T$ can be written as:
- $p(w_{1:T}) = \int_{\theta \in \Theta} p(w_{1:T}|\theta)p(\theta)\, d\theta$

In-context learning

Downstream task data distribution p_{prompt} :
(Xie et al., 2022)

$p_{prompt}(h_0)$



- All examples from the same task share the same latent concept variable θ^*.
- Let $x = w_{1:T-1}$ denote the input text sequence and $y = w_T$ denote the discrete output label.
- Only the distribution of h_0 changes to p_{prompt}, all the transition probabilities $p(h_i|h_{i-1})$ and the emission probabilities $p(w_i|h_i)$ remains the same as pre-training distribution.

In-context learning

- Let w^{delim} be a special delimiter token (e.g. white space or new line) to separate few-shot demonstrations in the in-context learning prompt.
- Let (x_{test}, y_{test}) be a test example.
- K-shot in-context learning prompt: $x_1, y_1, w^{delim}, x_2, y_2, w^{delim}, \dots, x_k, y_k, w^{delim}, x_{test}$
- Note that the above prompt distribution is very different from the pre-training distribution.
- Denote the k demonstrations by $S_k = x_1, y_1, w^{delim}, x_2, y_2, w^{delim}, \dots, x_k, y_k, w^{delim}$
- **Want**: the in-context learning predictor $argmax_{y \in V}\ p(y|S_k, x_{test})$ approaches to the Bayes optimal predictor $argmax_{y \in V}\ p_{prompt}(y|x_{test})$ as the number of in-context learning demonstrations grows.

In-context learning

- To show the previous statement, we need to assume:
  - In the pre-training data distribution, the delimiter token w^{delim} corresponds to a fixed set of hidden state values $D \subset H$.
  - The delimiter token w^{delim} is almost equally likely (with bounded probability) to appear anywhere in a token sequence.
  - The prompt distribution of the first hidden state $p_{prompt}(h)$ is close to the pre-training data distribution of the first hidden state $p(h|h^{delim}, \theta^*)$. In this case, the delimiter token acts as a restart token under the pre-training data distribution.
  - $p(S_k, x_{test}|\theta^*) > 0$

In-context learning

- We can expand the in-context learning predictor as follows:

$$
\begin{aligned}
p(y|S_n, x_{test}) &= \int_\theta p(y|S_n, x_{test}, \theta) p(\theta|S_n, x_{test}) d\theta \\
&\propto \int_\theta p(y|S_n, x_{test}, \theta) p(S_n, x_{test}|\theta) p(\theta) d\theta \qquad \text{(Bayes' rule, drop the constant } \frac{1}{p(S_n, x_{test})}\text{)} \\
&= \int_\theta \sum_{h_1 \in \mathcal{H}} p(y|x_{test}, h_1, \theta) p(h_1|S_n, x_{test}, \theta) \frac{p(S_n, x_{test}|\theta)}{p(S_n, x_{test}|\theta^*)} p(\theta) d\theta \\
&\qquad \text{(Law of total prob, Markov property, divide by } p(S_n, x_{test}|\theta^*) \text{ (a constant))}
\end{aligned}
$$

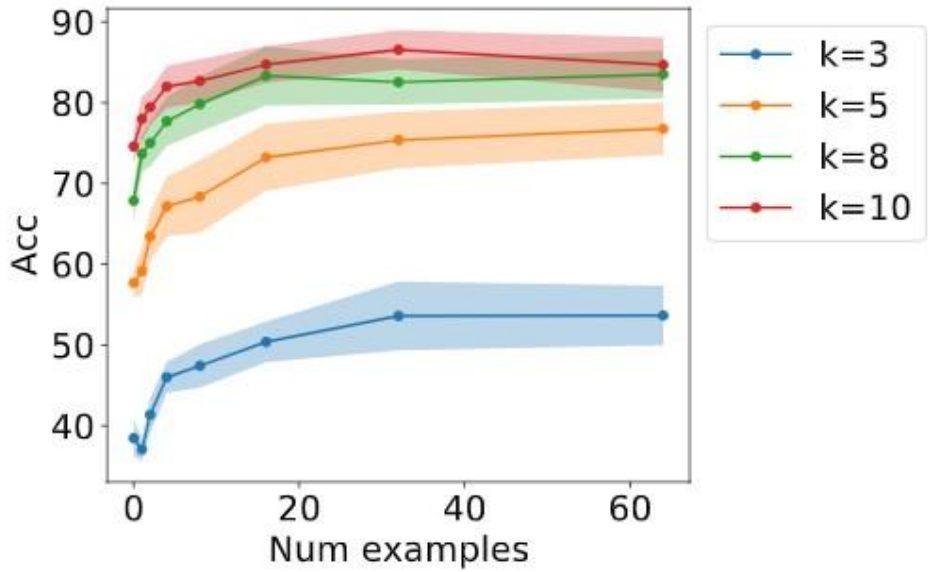
- Then we can prove that an LLM implicitly selects the correct latent concept variable value (omit the technical details here):

$$
\forall \theta \neq \theta^*, \lim_{n \to +\infty} \frac{p(S_n, x_{test}|\theta)}{p(S_n, x_{test}|\theta^*)} = 0
$$

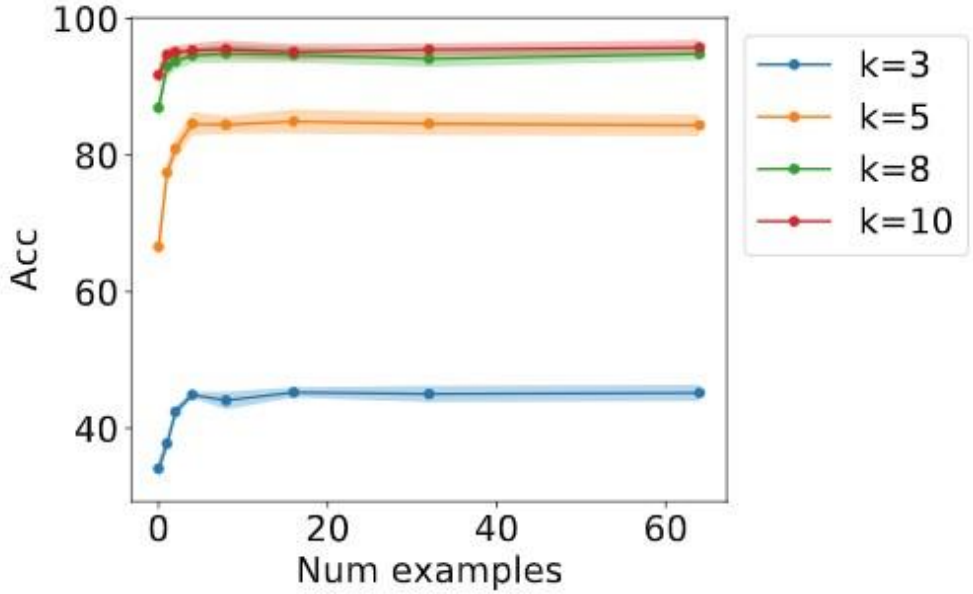
- This implies the intended conclusion:

$$
\lim_{n \to +\infty} \arg\max_{y \in \mathcal{V}} p(y|S_n, x_{test}) = \arg\max_{y \in \mathcal{V}} p_{prompt}(y|x_{test})
$$

In-context learning



Transformer



LSTM

- Experiments on synthetic data using a GPT-like language model.
- Vocabulary size = 150. Length of a train example = k. Length of a test example uniformly sample in [2,k].

Future directions and current progress

Comments and future directions

- Most works on understanding LLMs are not intended to open the black box. Instead, they try to get around the internal mechanism of LLMs by assuming they can perfectly estimate the pre-training distribution.
- In reality, LLMs systematically underestimate the rare word sequences (LeBrun, 2022). While human language distribution is heavy-tailed in nature, this means a large portion of possible word sequences is underestimated. A similar idea has been used to detect machine-generated text (Mitchell et al., 2023).
- There is a gap between the theoretical/empirical results derived with synthetic data, and the real-world LLM behavior. E.g. Language distribution is not HMM, we cannot have infinite demonstrations in a prompt, etc. There is no guarantee the derived results can be generalized to the real-world scenario.
- There are also contradicting conclusions in the current literature. e.g. Xie et al. (2022) show that LSTM can do in-context learning while Chan et al. (2022) show only Transformer can do in-context learning. Min et al. (2022) show that ground truth labels do not matter for demonstrations while Yoo et al. (2022) show that ground truth labels matter.

Current progress

- **Goal**: closing the gap between theory and real-world LLMs.
- **Current progress**: a step on verifying the previously introduced latent concept variable model for in-context learning using GPTs.
  - *Large Language Models Are Implicitly Topic Models: Explaining and Finding Good Demonstrations for In-Context Learning*. Xinyi Wang, Wanrong Zhu, William Wang. Preprint 2023.

LLMs are implicitly topic models

LLM: $P(\boldsymbol{w}_{1:T}) = \prod_{i=1}^{T} P(\boldsymbol{w}_i|\boldsymbol{w}_{i-1}, ..., \boldsymbol{w}_1)$ **Topic model:** $P(\boldsymbol{w}_{1:T}) = \int_{\Theta} P(\boldsymbol{w}_{1:T}|\boldsymbol{\theta})P(\boldsymbol{\theta})d\boldsymbol{\theta}$

Language model probability output by an LLM

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

Generated continuation

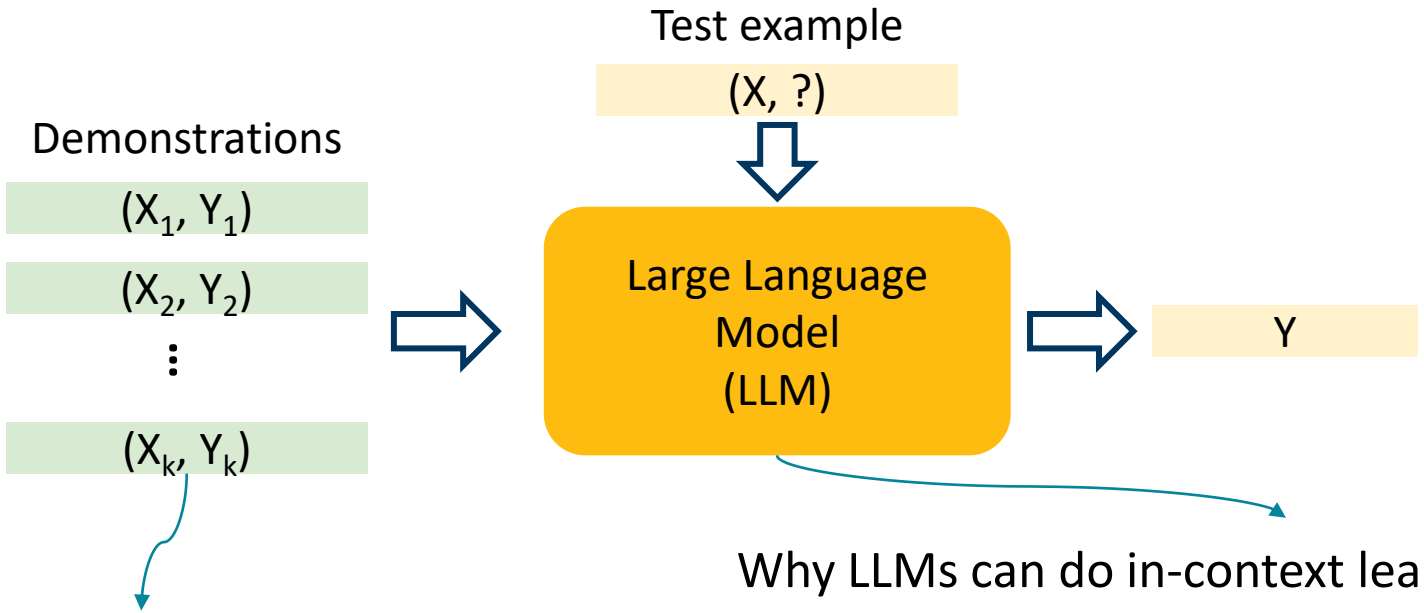
Prompt

LLMs generate the continuation exclusively based on the inferred concept variable θ

LLMs implicitly infer a latent concept variable θ from the prompt

- Assumption: the generated continuation is independent of the prompt given the concept variable θ.

In-context learning



In-context learning is highly unstable:

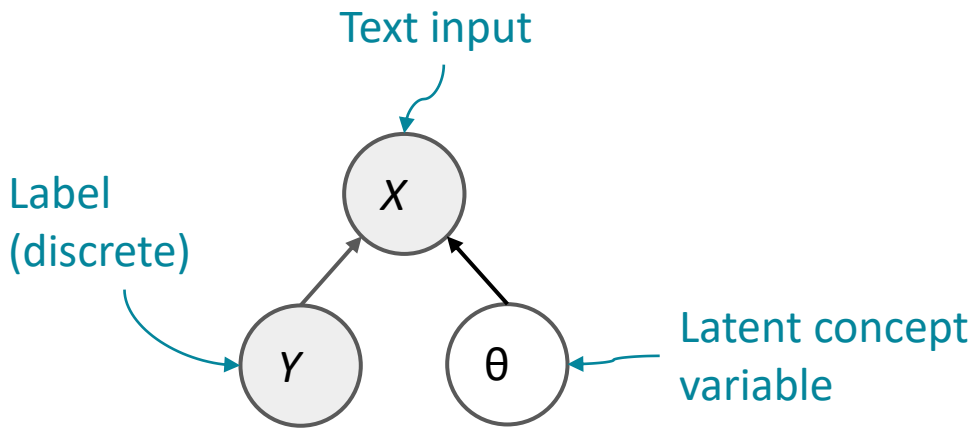
- How do we choose the demonstrations if we have a set of annotated data?
  Similarity? (Liu et al. 2022; Su et al. 2022)
  Entropy of predicted labels? (Lu et al. 2022)

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)
- Smaller models encoded in activation? (Akyurek et al. 2022)

- How can we understand in-context learning in a real-world setting?

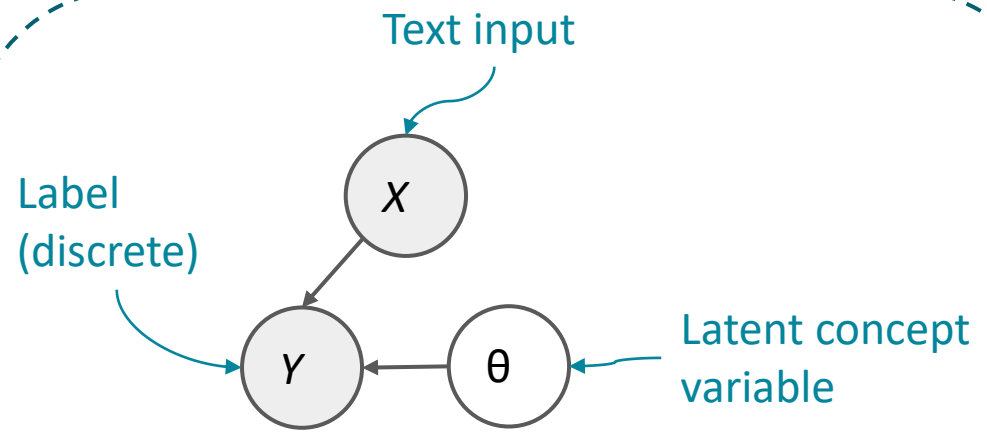
Data generation direction matters



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

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

Bayes optimal classifier



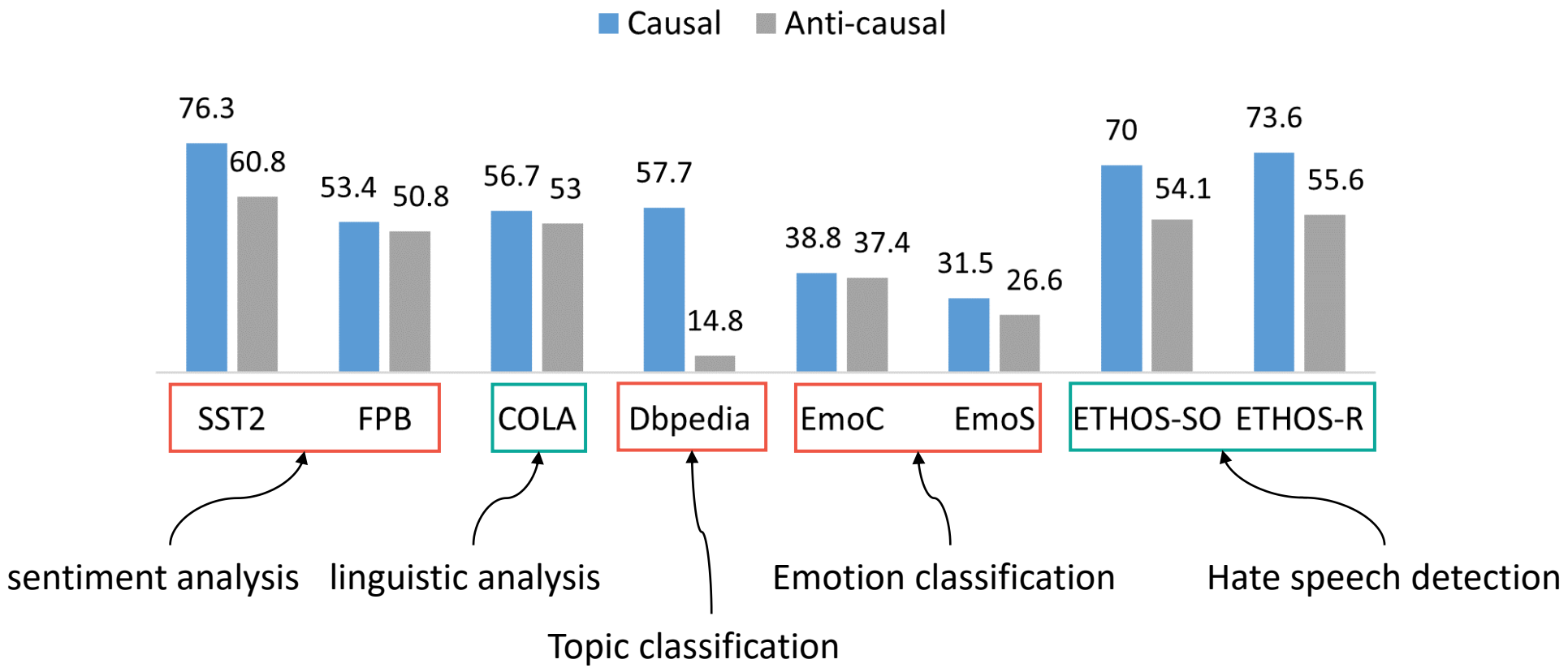
e.g. linguistic analysis, hate speech detection

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

Bayes optimal classifier

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

Causal v.s. anti-causal



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

Analysis in-context learning classifier

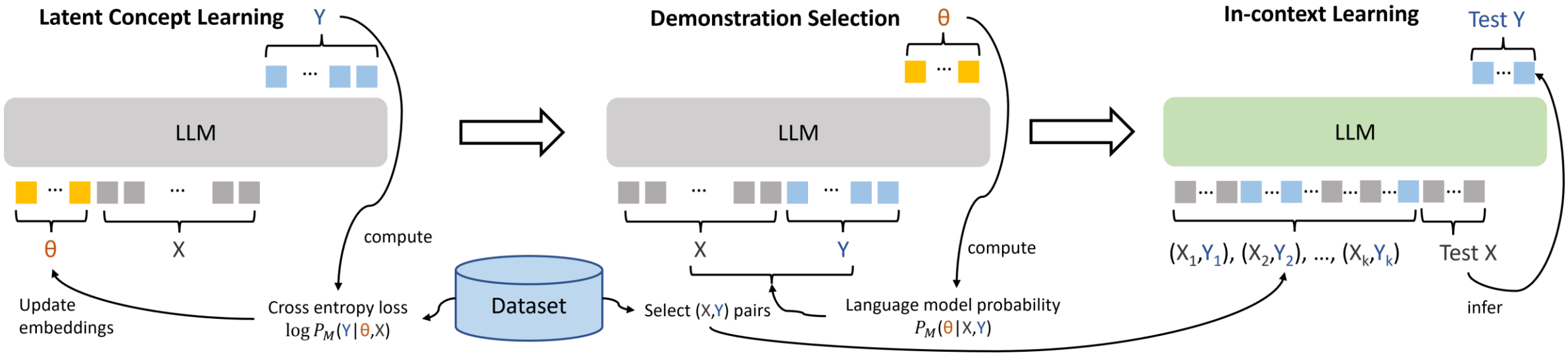
$$P_M^d(Y|X_1^d, Y_1^d, ..., X_k^d, Y_k^d, X)$$
$$= \int_\Theta P_M^d(Y|\boldsymbol{\theta}, X) P_M^d(\boldsymbol{\theta}|X_1^d, Y_1^d, ..., X_k^d, Y_k^d, X) d\boldsymbol{\theta}$$

Latent concept variable learning (soft prompt tuning)

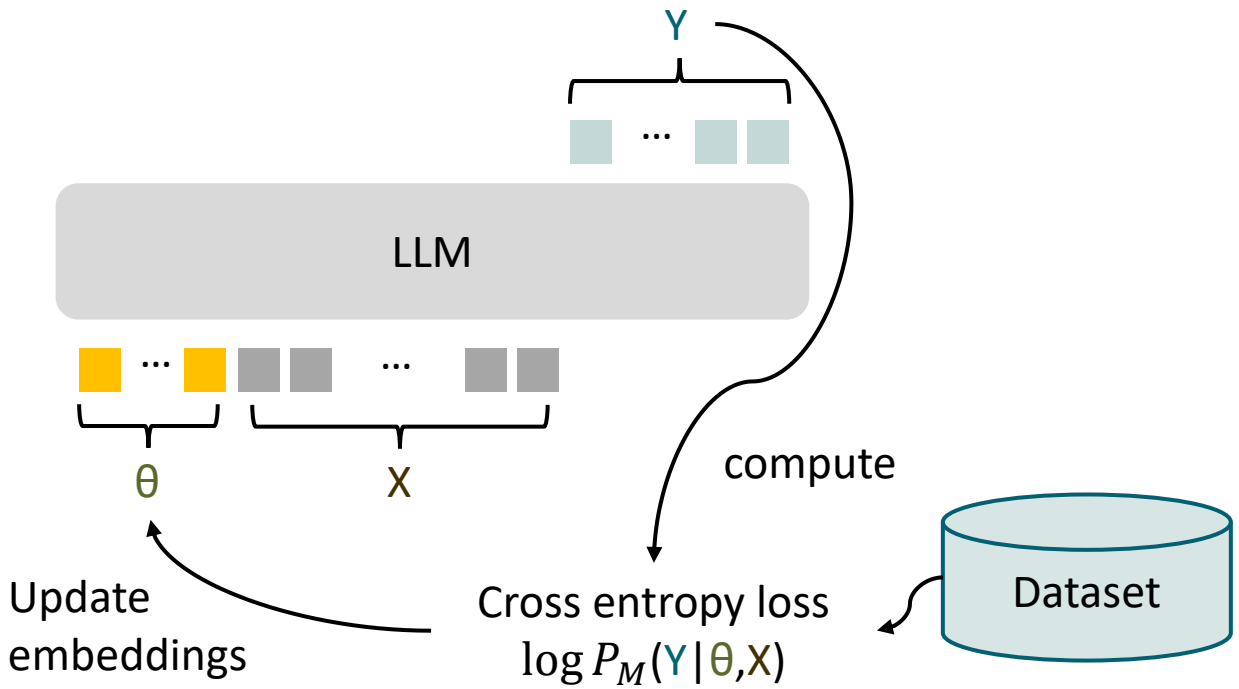
Demonstration selection

- We want to make the above in-context learning classifier $P_M^d(Y|X_1^d, Y_1^d, ..., X_k^d, Y_k^d, X)$ as close to the Bayes optimal classifier as possible, which means we need to make $P_M^d(\boldsymbol{\theta}|X_1^d, Y_1^d, ..., X_k^d, Y_k^d, X)$ as concentrated on the θ 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

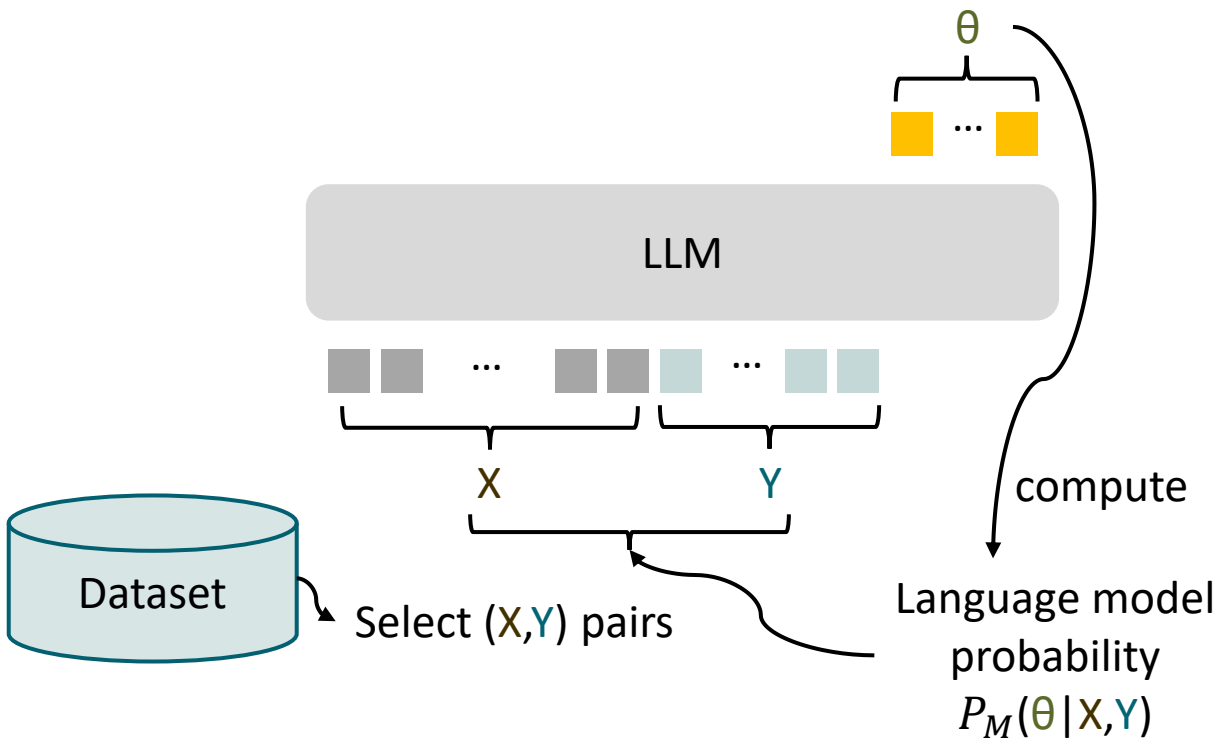


Latent Concept Learning



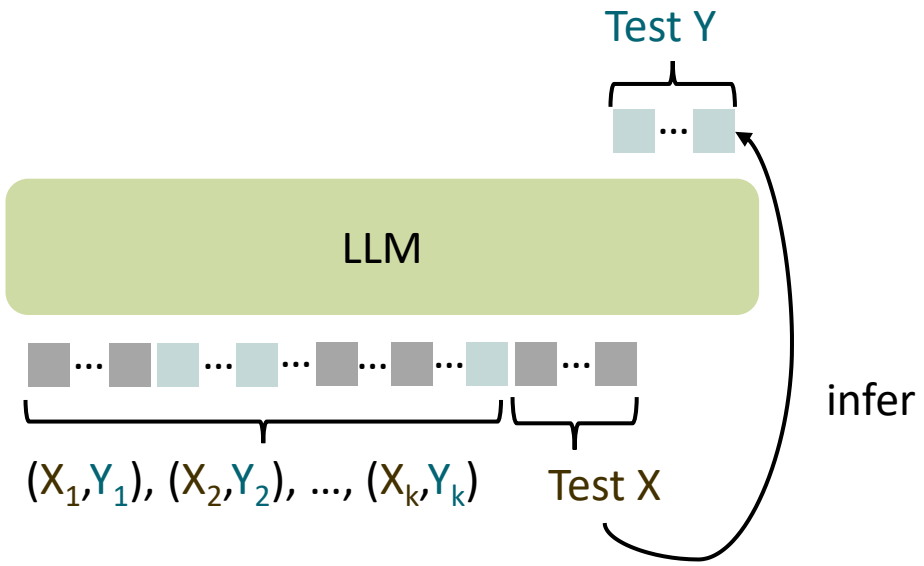
- Add a few new concept tokens to the original vocabulary of the LLM.
- Train the embedding of these concept tokens while freezing all other parameters, such that the LLM can predict the label Y given X and the concept tokens as prefixes.
- Use GPT2-large in practice.

Demonstration Selection



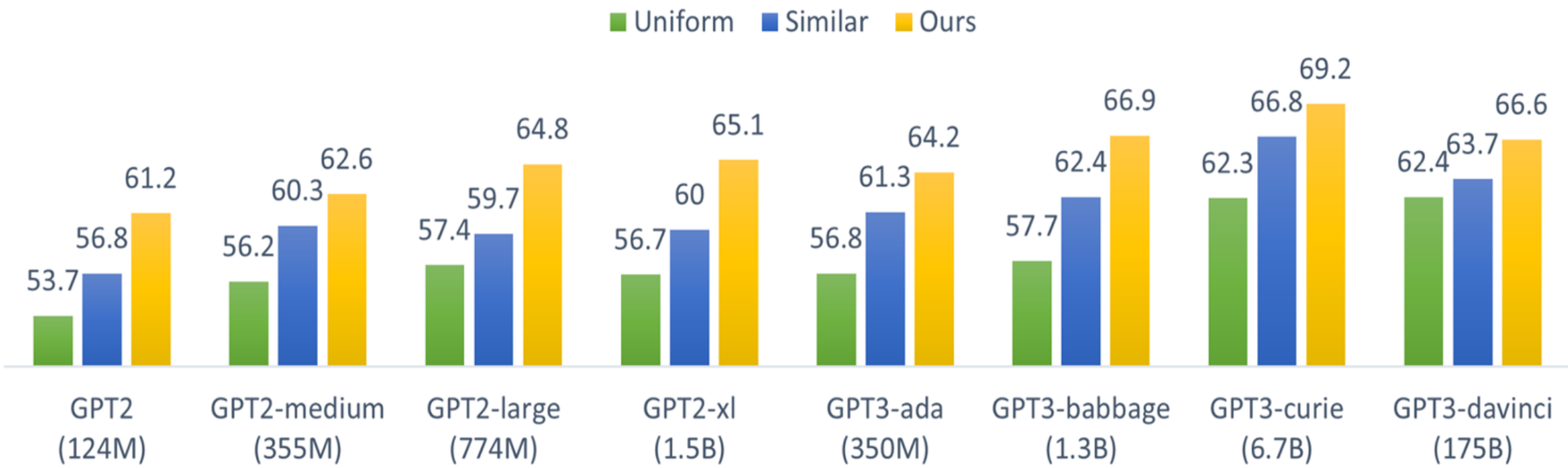
- Compute the LM probability of predicting the concept tokens given an example (X, Y).
- Then choose the top-k examples producing the highest probabilities as the demonstrations for in-context learning.
- Use GPT2-large in practice.

In-context Learning



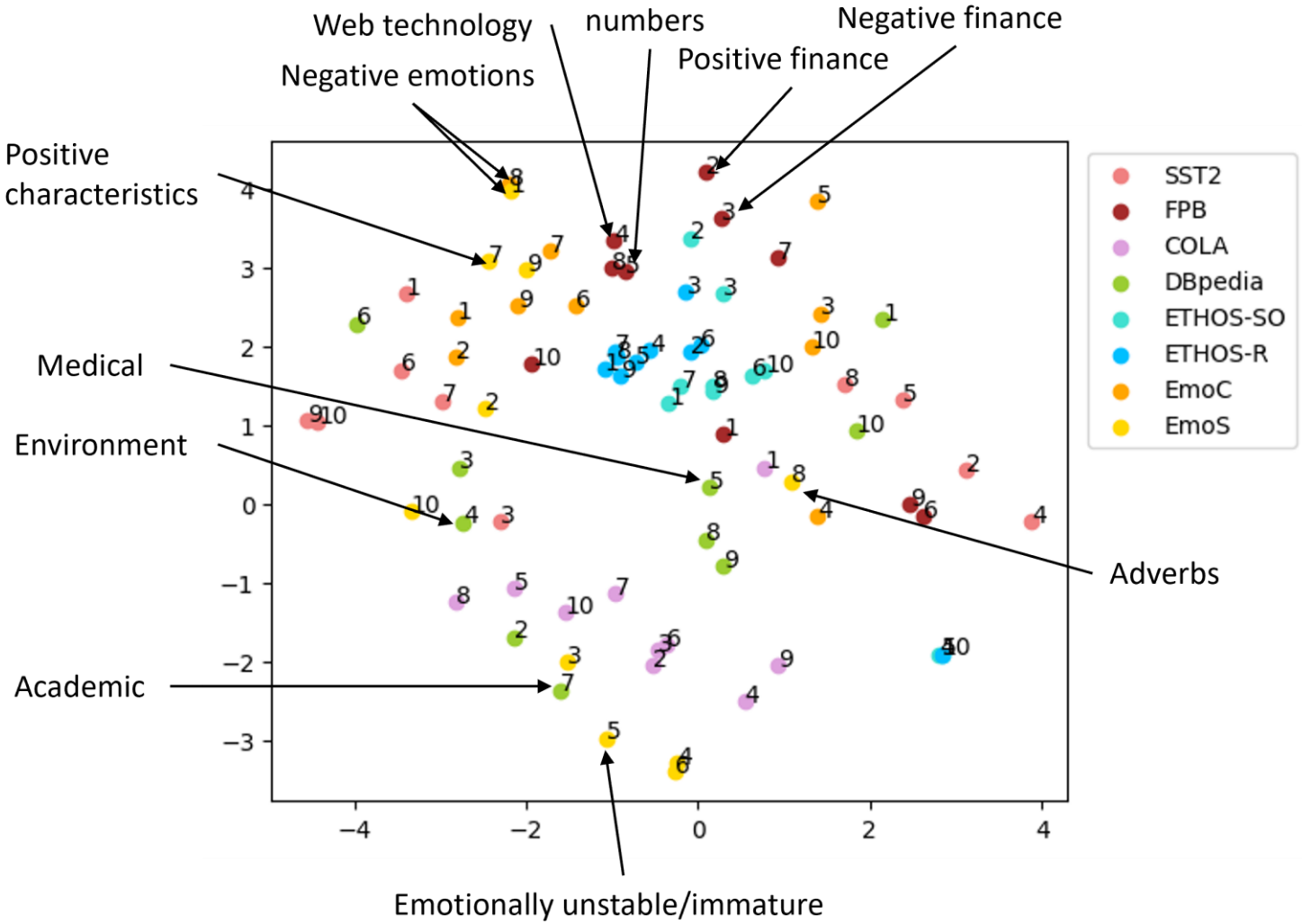
- Test the performance of the chosen k demonstrations by using them for in-context learning on a separate test set.
- Different LLMs from the previous stages can be used.
- Use different sizes GPTs in practice.

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

Conclusions

- 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.

Thank you!

Questions?