5/5/2023

Understanding Pre-trained Large Language Models through a Probabilistic Lens

Xinyi Wang

Outline

- Background on large language models
- Recent works on understanding large language models
- Future directions and my 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

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

Topic model

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

Language Model

Counting

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

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

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

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

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

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

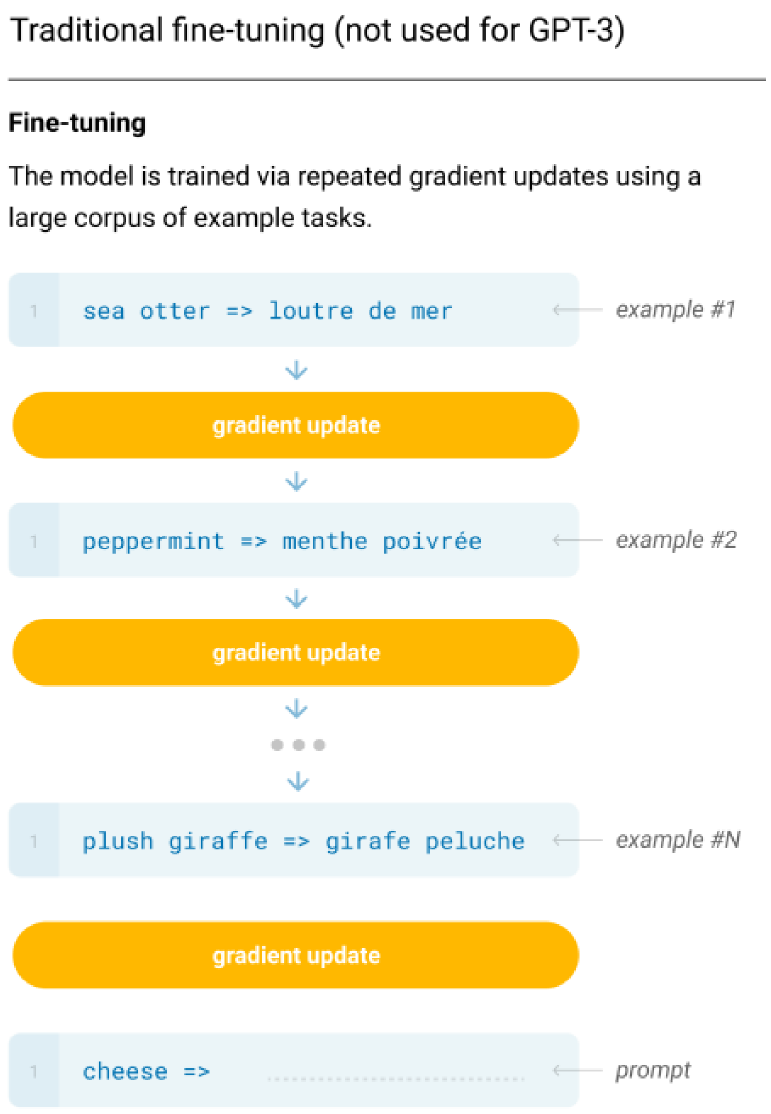
Beyond probability estimation

- 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

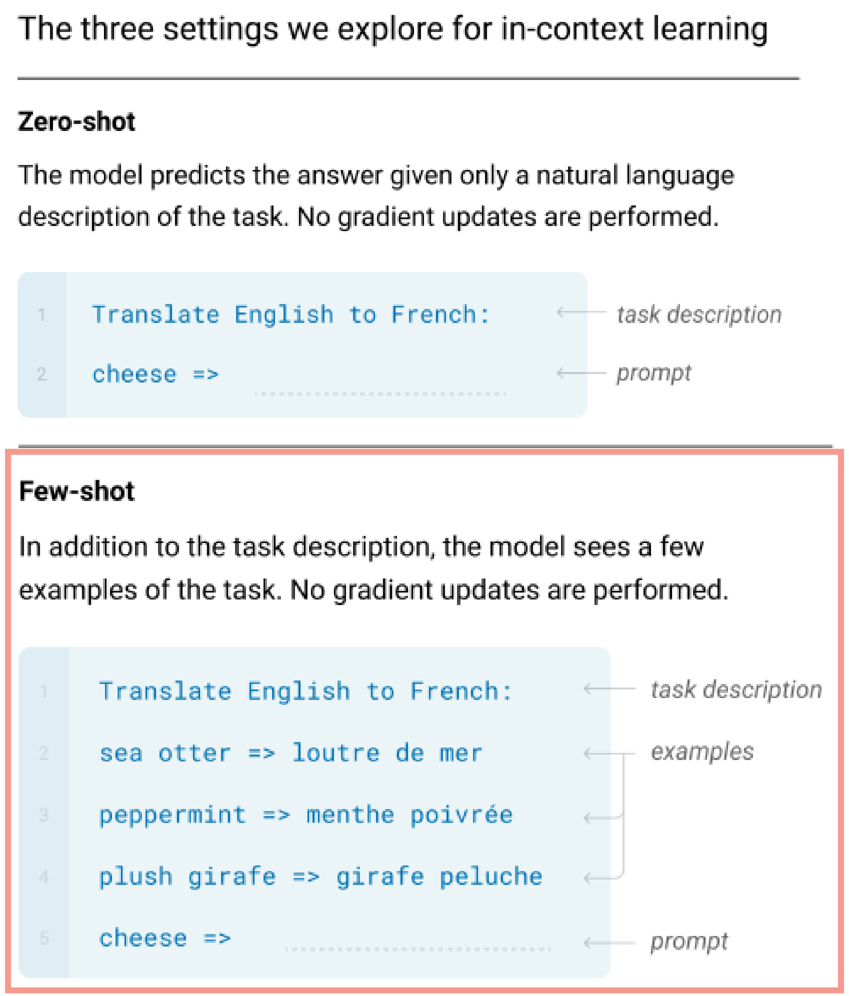
- **BERT (Devlin et al., 2018)**: a pre-trained masked language model, using an encoder-only Transformer architecture.
    - Serves as a good initialization for many downstream NLP tasks.
    - SOTA performance (at that time) on many NLP tasks can be achieved by fine-tuning BERT on corresponding training 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



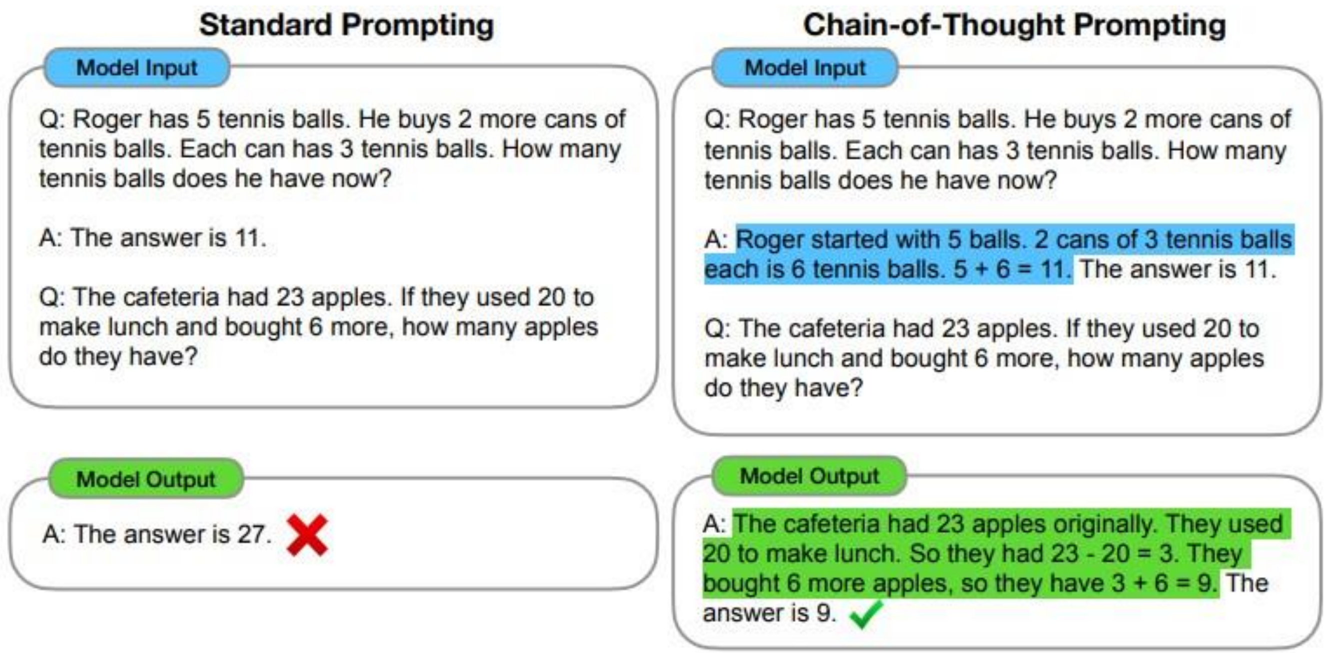
- Use the pre-trained large language model as a good starting point for learning downstream NLP tasks.
- Expensive to train when the model is large.
- Training data required (not necessarily a large amount).
- Parameter efficient fine-tuning: only tune a small number of parameters in the model and fix 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



(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

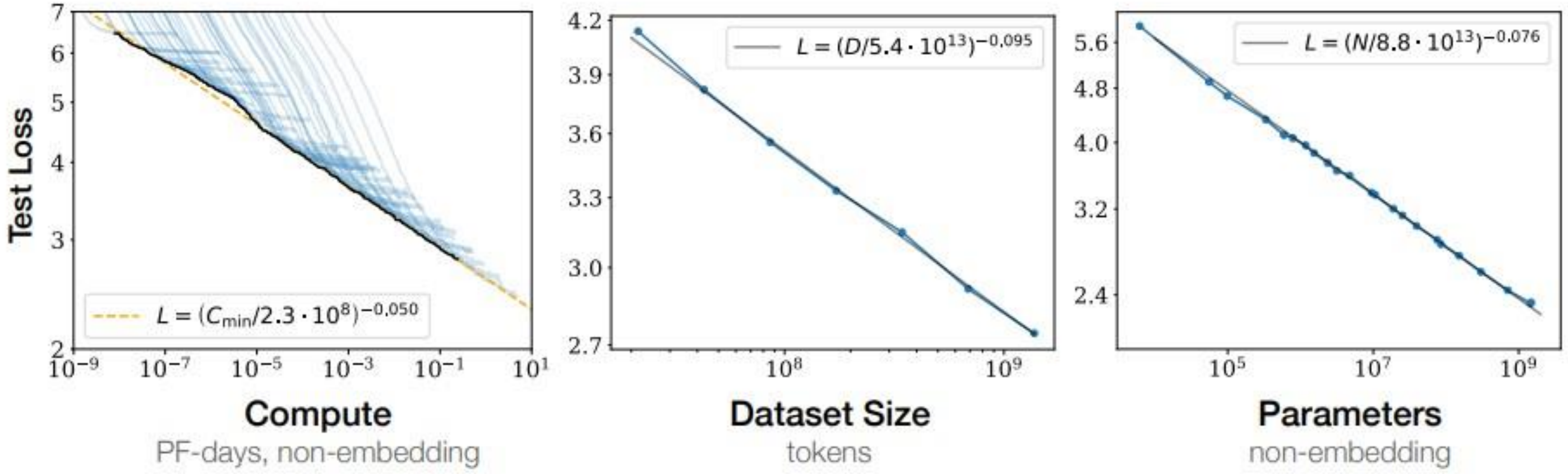
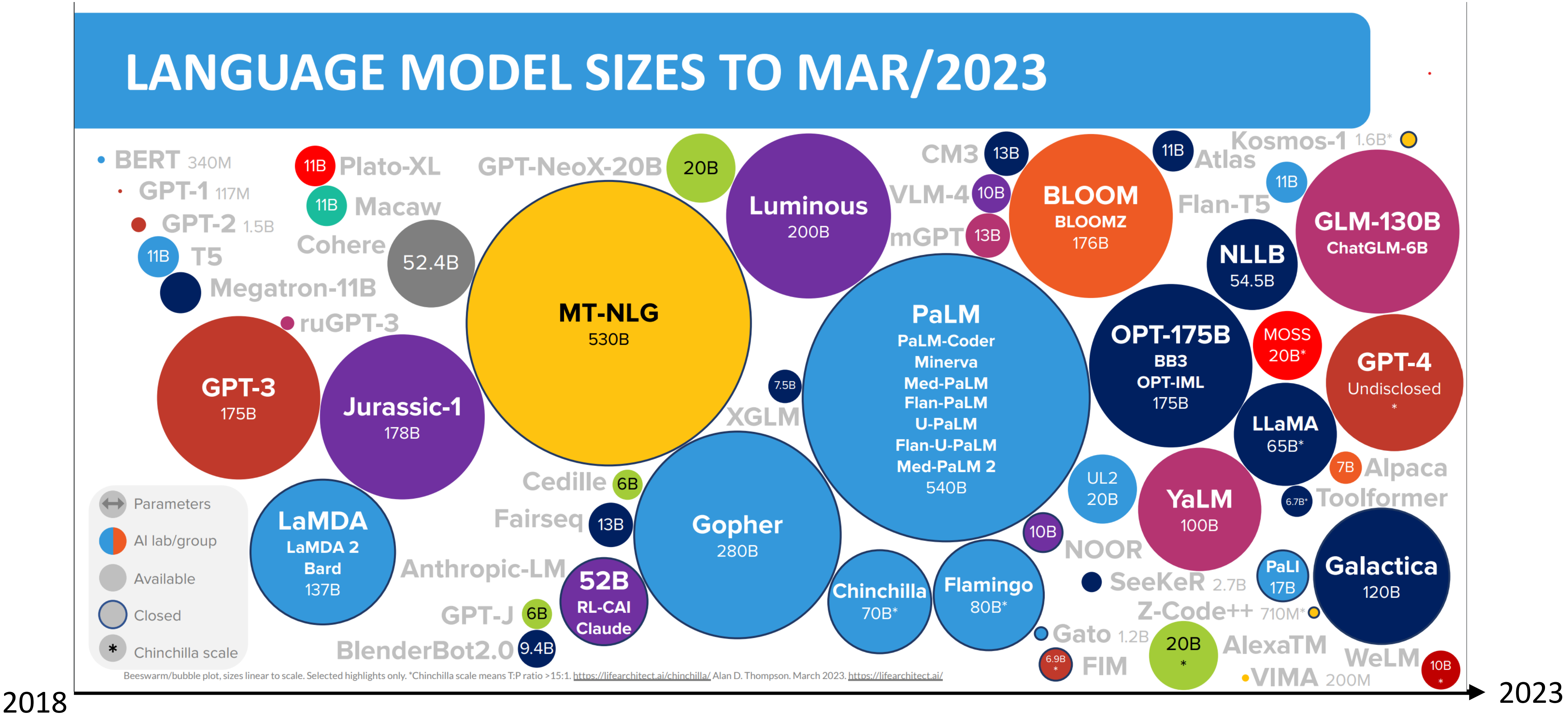


Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when 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.

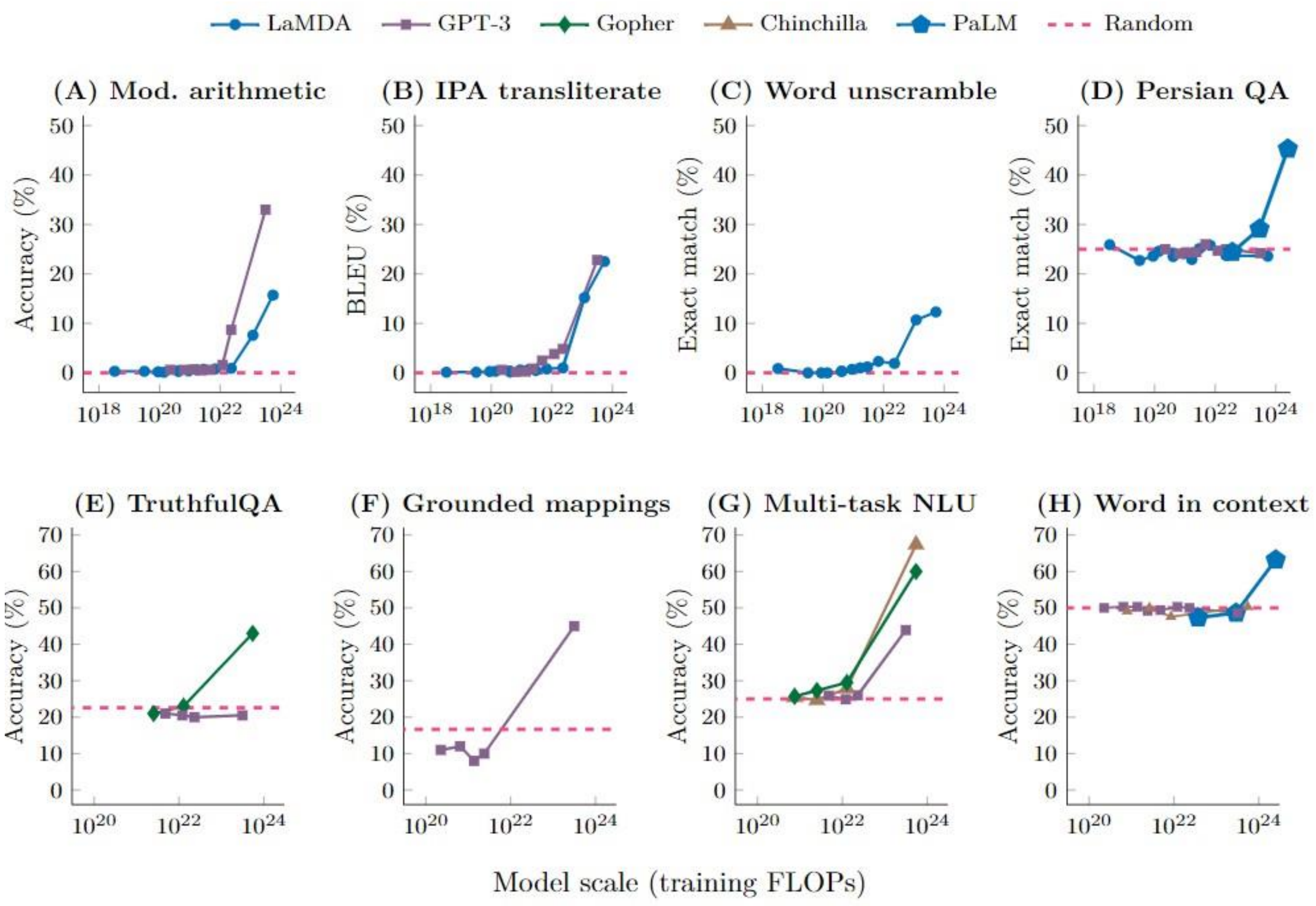
Existing large language models



- 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.



UC SANTA BARBARA

Recent works on understanding large language models

How to understand these phenomenon?

- 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. GPT4).
- Two main directions on understanding LLMs or Transformers:
  - **Mechanical**: Some basic matrix and computer operations can be exactly constructed with a Transformer (Lindner et al. 2023, Giannou et al. 2023)
  - **Bayesian**: LLMs are implicitly inferring a latent variable from the prompt (Jiang et al. 2023)
- Two ways to empirically verify a proposed theory:
  - Create **synthetic data** and pre-train a toy Transformer to perform experiment in a controlled environment (Pros: easy to control. Cons: Not sure if can be applied to real LLMs.)
  - **Directly verify** on real-world LLMs by design smarter experiments. (Cons: hard to control. Hard to prove the exact point. Pros: confirmed to explain LLMs.)

Understanding fine-tuning

- **Bayesian:**
  - **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: 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 (Wei et al. NeurIPS 2021).
    - **Experiments**: data generated from a synthetic distribution.
- **Mechanical**: ?

Understanding in-context learning

- **Bayesian**: examine pre-training data distribution
  - **Hidden Markov Model** (Xie, et al.).
  - **Compositional Attribute Grammar**: language can be mapped to trees (Hahn et al. 2023)
  - **skewed Zipfian distribution**: Burstiness, long tail, the dynamic meaning of words, etc (Chan et al., 2022).
  - The **unambiguity** of language (Jiang et al. 2023).
  - **Experiments**: data generated from a synthetic distribution.
- **Mechanical**: how LLMs utilizing the few-shot demonstrations
  - Mimic gradient descent at inference time (von Oswald et al. 2022, Dai et al. 2023)
  - Smaller models are encoded in activation (Akyurek et al. 2022)
  - Transformer itself is a learning algorithm (Li et al., 2023)
  - **Experiments**: data from a simple task. E.g. linear regression.
  - Dai et al. 2023 use pre-trained GPT2-like LLMs to verify their results.

Understanding 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).

Understanding emergent abilities

- **Bayesian**: the unambiguity of language + exact estimate of the marginal distribution of language (Jiang et al. 2023).
  - latent variable = intent of a message
  - Unambiguity = can exactly infer the correct intent of a message
  - Can explain why LLMs generate coherent continuations, do in-context learning, chain-of-thought prompting, instruction following
  - Problem: can LLMs precisely estimate the pre-train distribution? LeBrun et al. 2022 find that GPT2s systematically underestimate relatively rare text sequences, which constitute a significant portion of the long-tail distribution of language. A similar idea has been used to detect machine-generated text (Mitchell et al., 2023).
  - **Mechanical**: ?

Future directions and current progress

Comments and future directions

- Most works on understanding LLMs are not intended to open the black box of Transformers. Instead, they try to get around the internal mechanism of LLMs by assuming they can perfectly estimate the pre-training distribution.
- 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 latent concept variable model for in-context learning using real-world LLMs.
  - *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: $P_M(\underline{\boldsymbol{w}_{t+1:T}} | \underline{\boldsymbol{w}_{1:t}}) = \int_{\Theta} \underline{P_M(\boldsymbol{w}_{t+1:T} | \boldsymbol{\theta})} \underline{P_M(\boldsymbol{\theta} | \boldsymbol{w}_{1:t})} d\boldsymbol{\theta}$

Generated continuation

Prompt

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

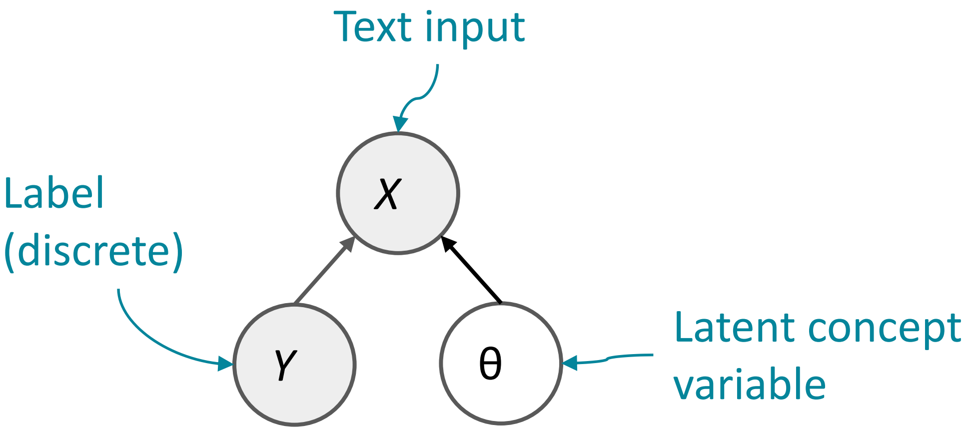
LLMs implicitly infer a latent concept variable θ from the prompt

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

In-context learning

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

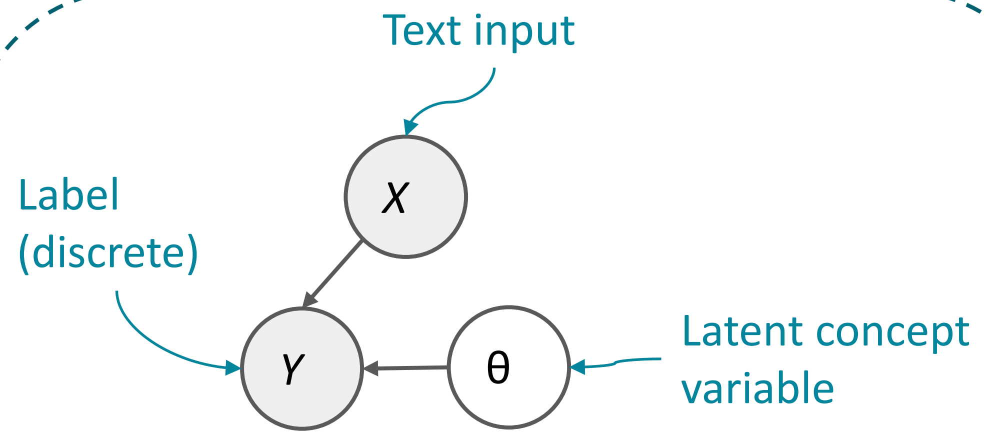
Data generation direction matters



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

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

Bayes optimal classifier



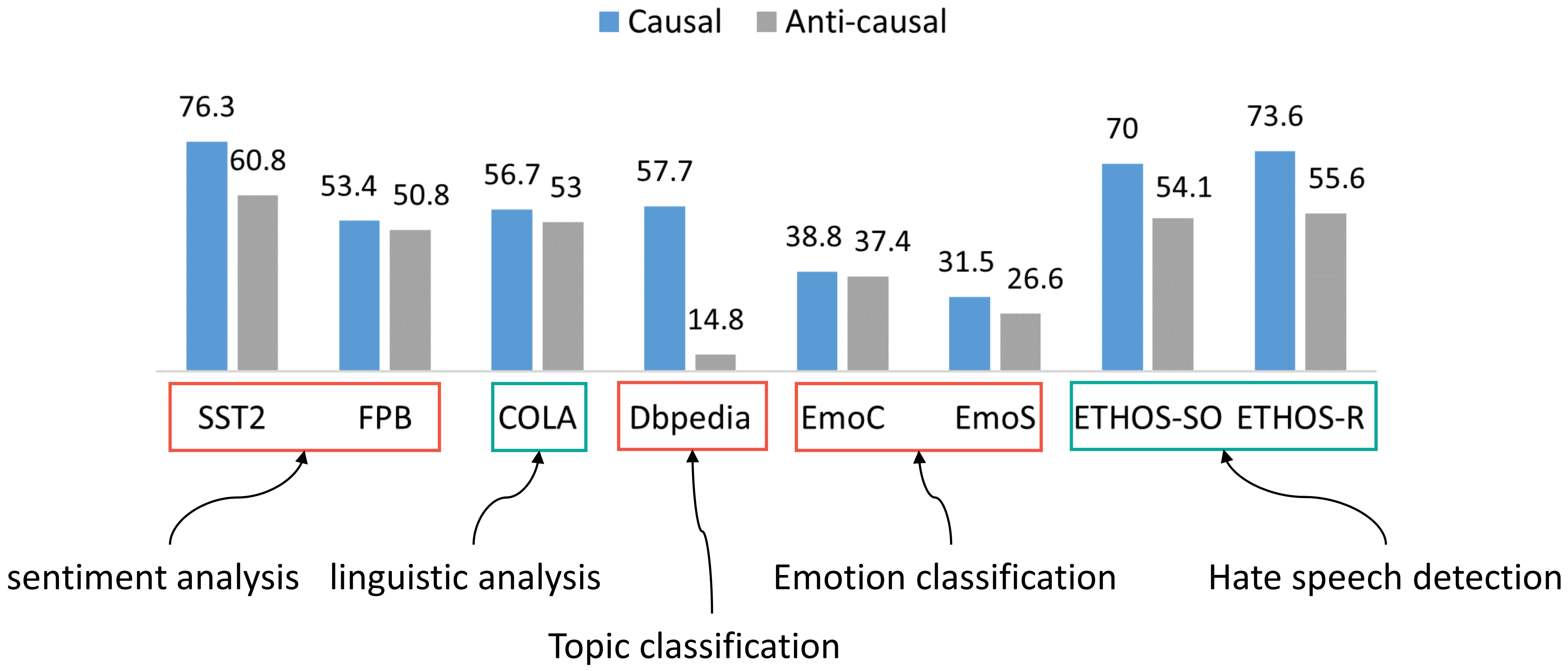
e.g. linguistic analysis, hate speech detection

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

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



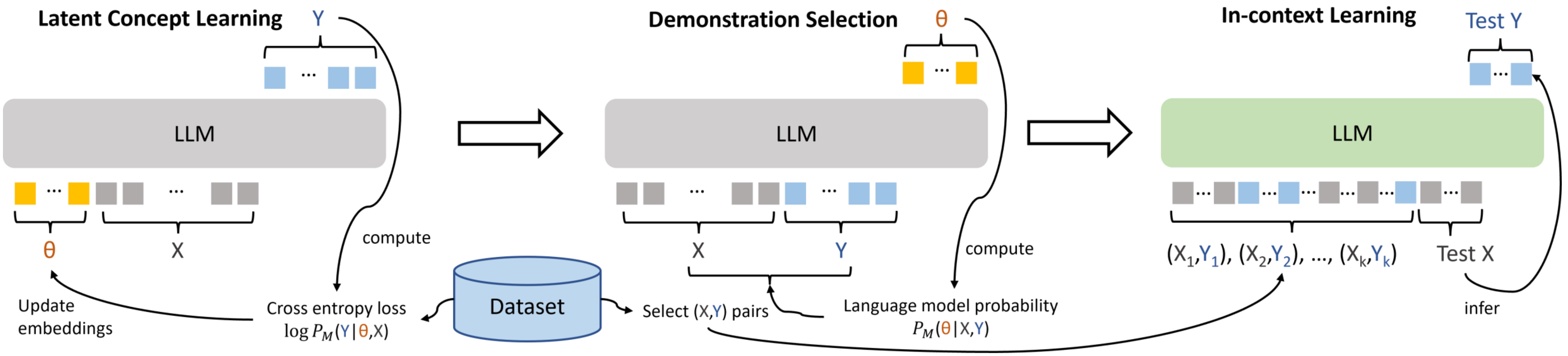
- 4-shot in-context learning accuracy with GPT2-large.

Analysis in-context learning classifier

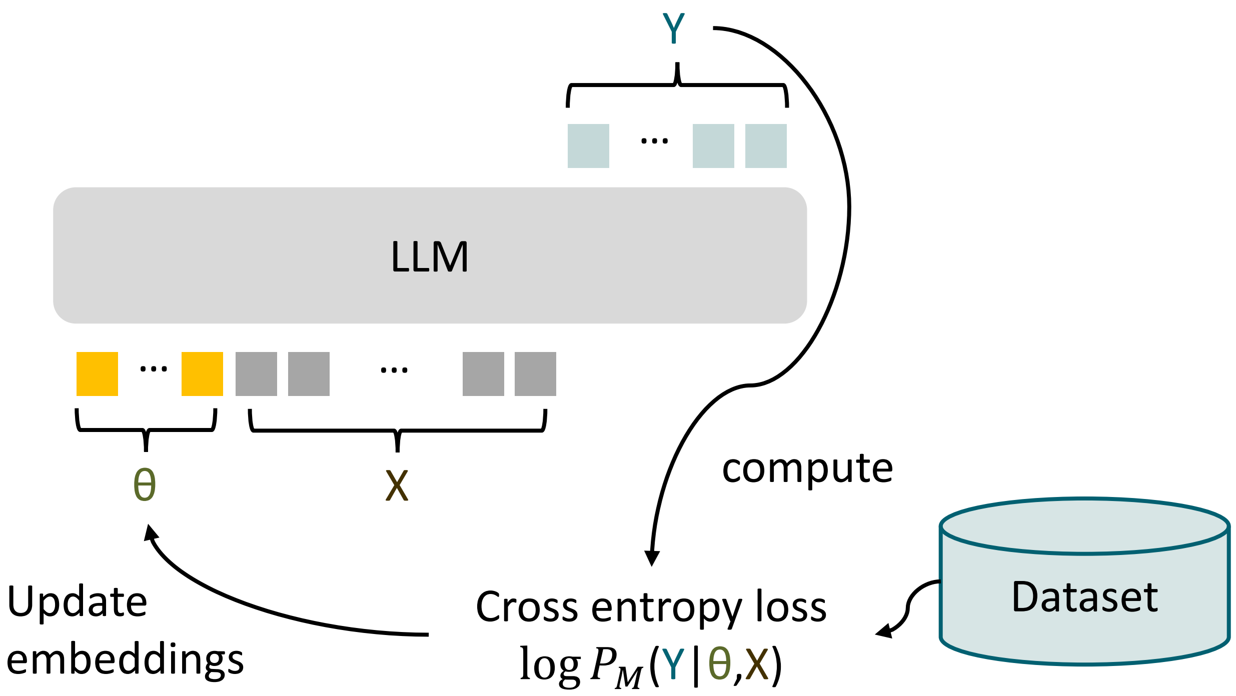
$$P_M^d(Y|X_1^d, Y_1^d, ..., X_k^d, Y_k^d, X)$$
$$= \int_\Theta \underbrace{P_M^d(Y|\boldsymbol{\theta}, X)}_{\text{Latent concept variable learning (soft prompt tuning)}} \underbrace{P_M^d(\boldsymbol{\theta}|X_1^d, Y_1^d, ..., X_k^d, Y_k^d, X)}_{\text{Demonstration selection}} d\boldsymbol{\theta}$$

- 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 optimal θ value corresponding to task d as possible.
- We can use the above conclusion to first learn a delegate of the optimal latent value, and then use the delegate to choose the best demonstrations from a set of annotated data.

Algorithm overview

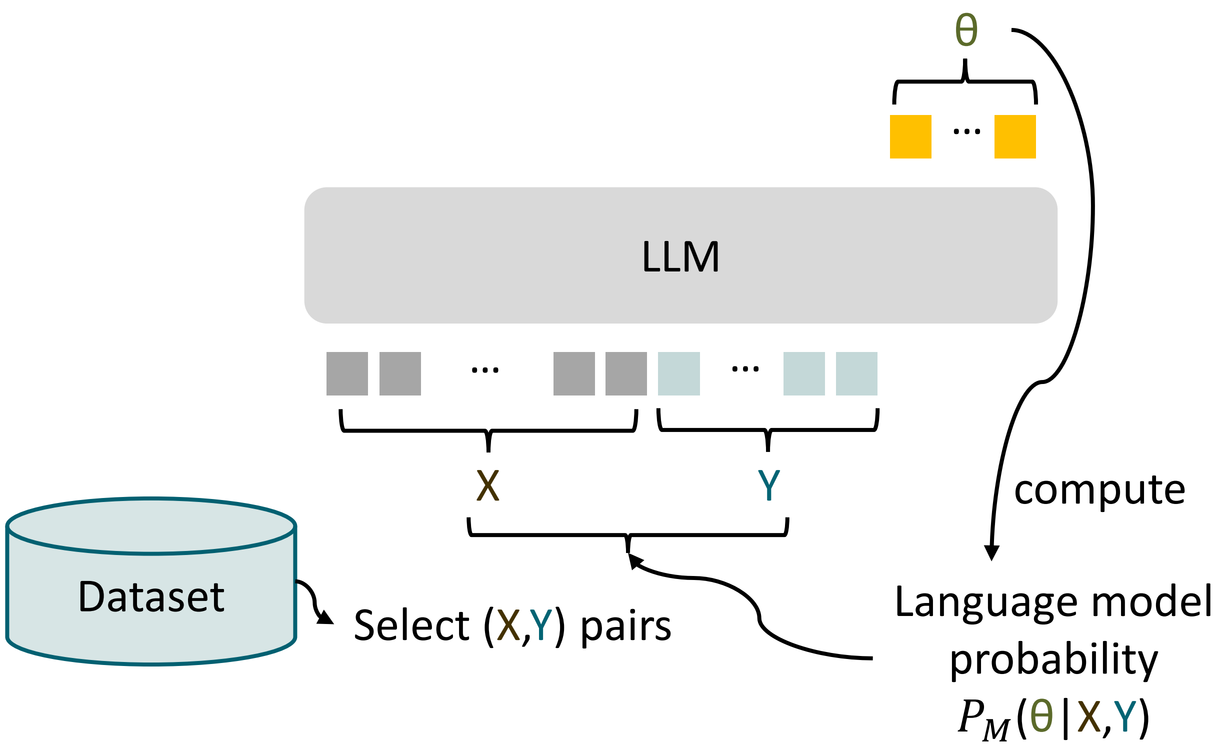


Latent Concept Learning



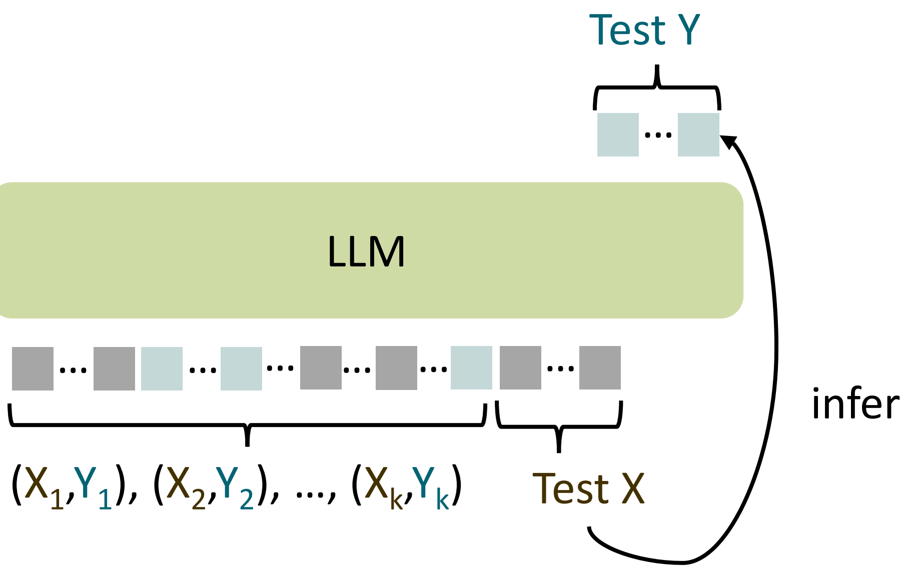
- Add a few new concept tokens to the original vocabulary of the LLM.
- Train the embedding of these concept tokens while freezing all other parameters, such that the LLM can predict the label Y given X and the concept tokens as prefixes.
- Use GPT2-large in practice.

Demonstration Selection



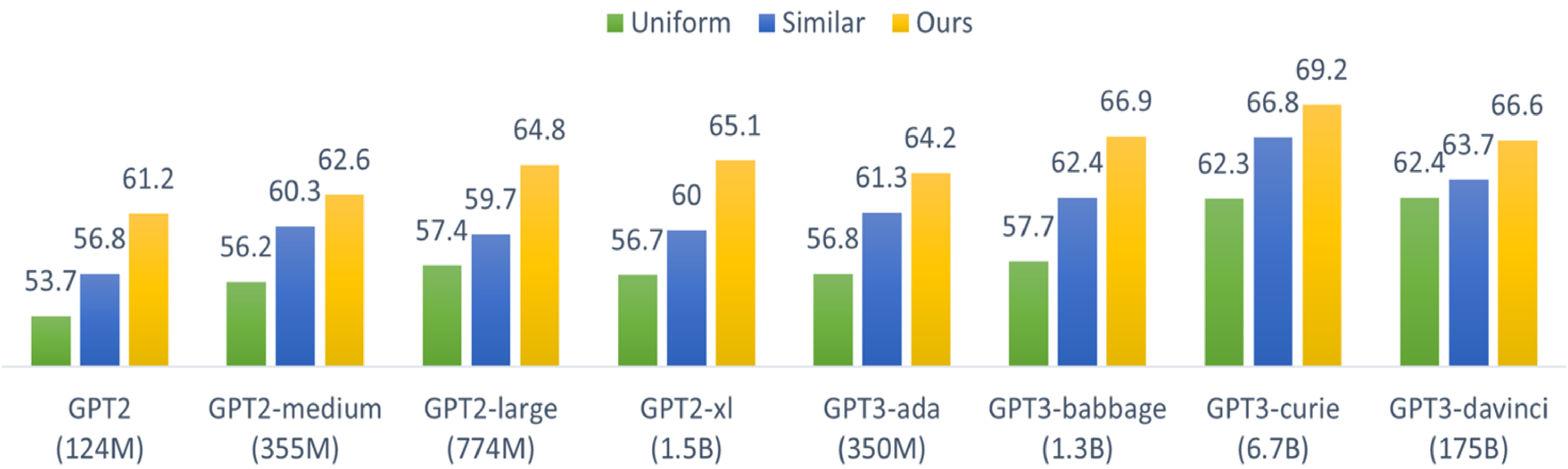
- Compute the LM probability of predicting the concept tokens given an example (X, Y).
- Then choose the top-k examples producing the highest probabilities as the demonstrations for in-context learning.
- Use GPT2-large in practice.

In-context Learning



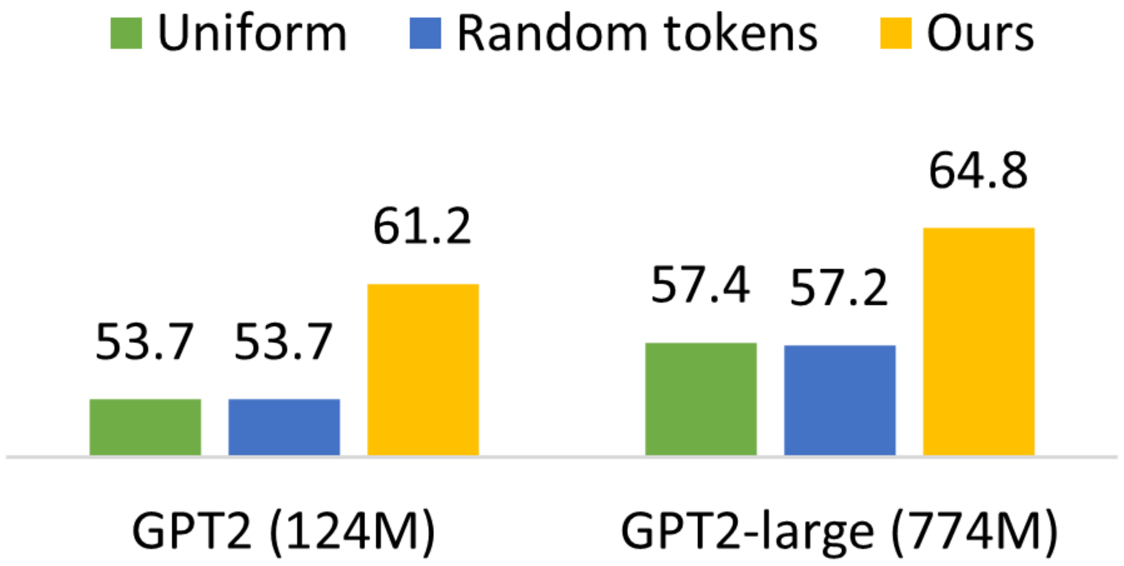
- Test the performance of the chosen k demonstrations by using them for in-context learning on a separate test set.
- Different LLMs from the previous stages can be used.

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.

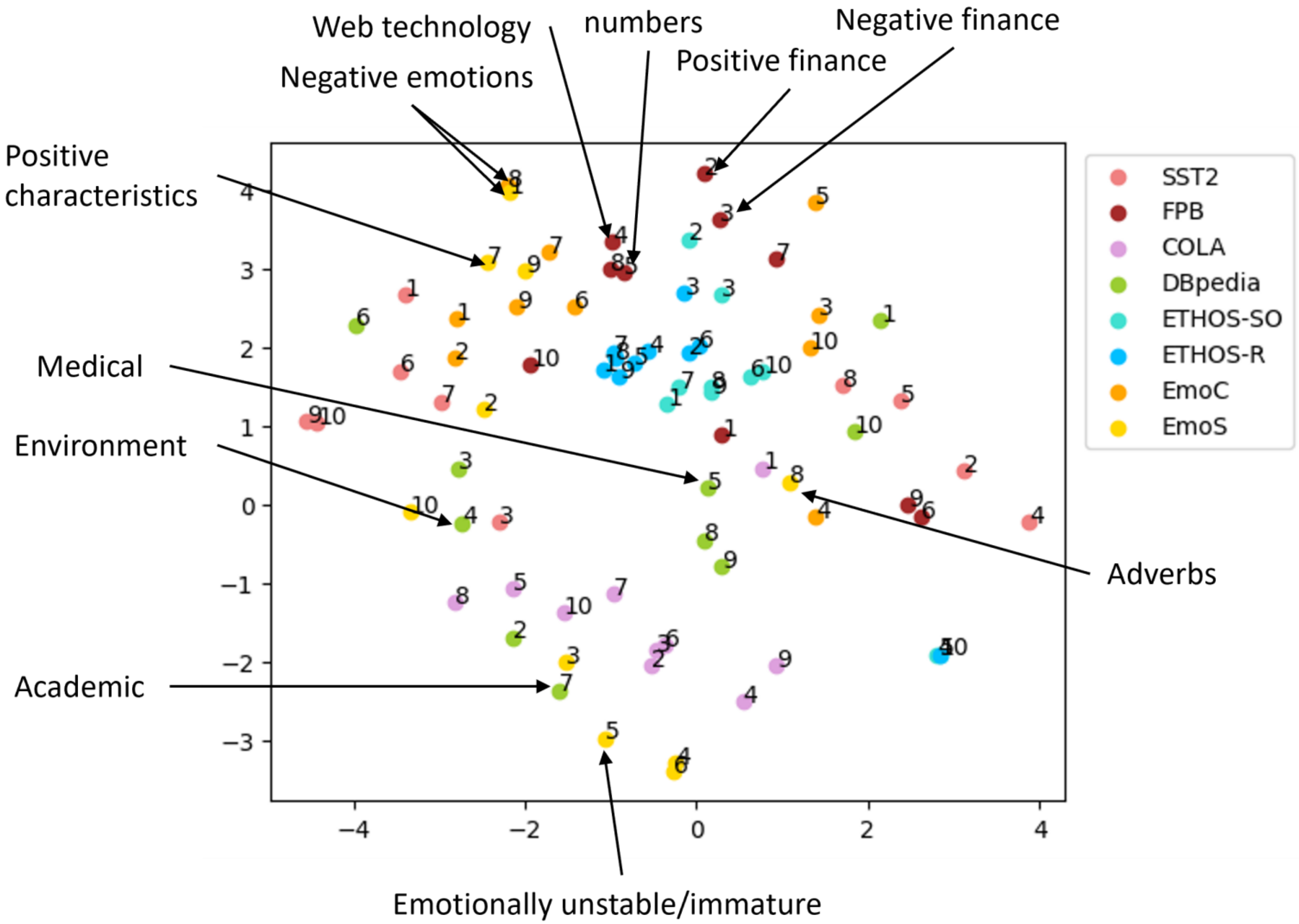
Does latent variable really help?



- Random tokens selected from the vocabulary are in place of the learned concept tokens for selecting demonstrations.
- 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.

UC SANTA BARBARA

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?