How does In-Context Learning work?

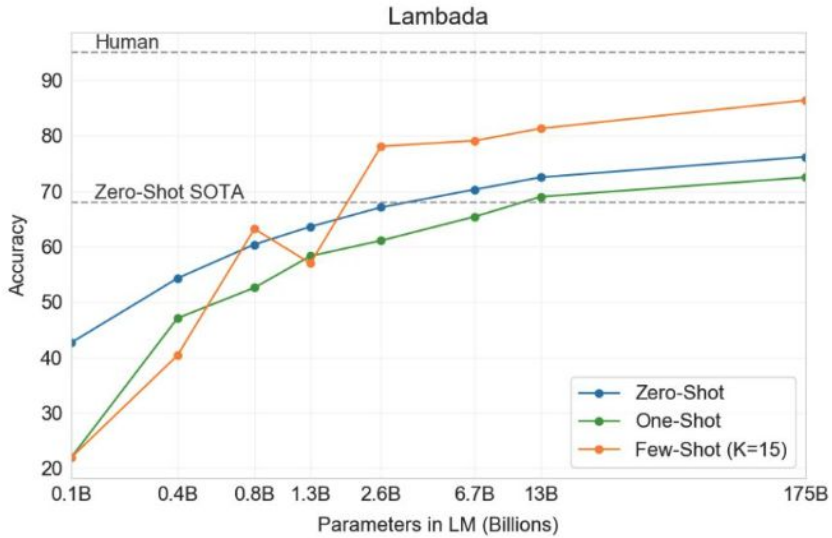
Based on ‘An Explanation of In-context Learning as Implicit Bayesian Inference’

Presented by: Mithilesh Vaidya

What is In-Context Learning (ICL)?

- Ability of a LM to complete a query **based** on input-output examples given in context
- Same sentence may have different concepts but LM figures the target underlying concept and predicts!

What can it do?



- Beats SOTA benchmarks for LAMBADA (commonsense sentence completion) and TriviaQA (question-answering)
- Beats models trained with supervision

Beyond benchmarks:

- Write code from natural language descriptions
- Generalizing spreadsheet functions (better FlashFill)

Why is it mysterious?

- **Task mismatch**: LLMs are pre-trained for next-token prediction ONLY; Model not explicitly pre-trained to learn from examples
- **No weight updates**/fine-tuning; everything computed and stored in forward pass!

From a human perspective, it still feels like next-token prediction. Why not for LLMs?

- LLMs pre-trained for next-token prediction on coherent data
- Within a context, no **abrupt** transitions
- (Pre-training distribution) != (Prompt distribution) due to abrupt low-probability transitions
- No encoder-decoder architecture to force LM to learn underlying concept

Example: Wiki bios

Pre-training text	Albert Einstein was a German-born theoretical physicist, widely acknowledged to be one of the greatest and most influential physicists of all time.
PT Structure	Name -> Nationality -> Occupation -> …
Prompt	Albert Einstein was German Mahatma Gandhi was Indian Marie Curie was ?
Prompt Structure	Name -> Nationality -> \n -> Name -> Nationality -> \n ….

Very low probability transitions:

- German -> Mahatma
- Indian -> Marie

Proposed framework

Step	Humans	LM
1	Observe all examples at once	Input: Prompt (= IO pairs with delimiters)
2	Extract **common** underlying **concept** from given examples	Infer/Locate $\boldsymbol{\theta}^*$ (latent) from prompt
3	Apply it to new example	$p(y_{test} \mid x_{test}, \theta^*)$

Note: This is a possible theory; many others such as meta-gradients
We study a toy dataset (and not real text)

Bayesian Inference view

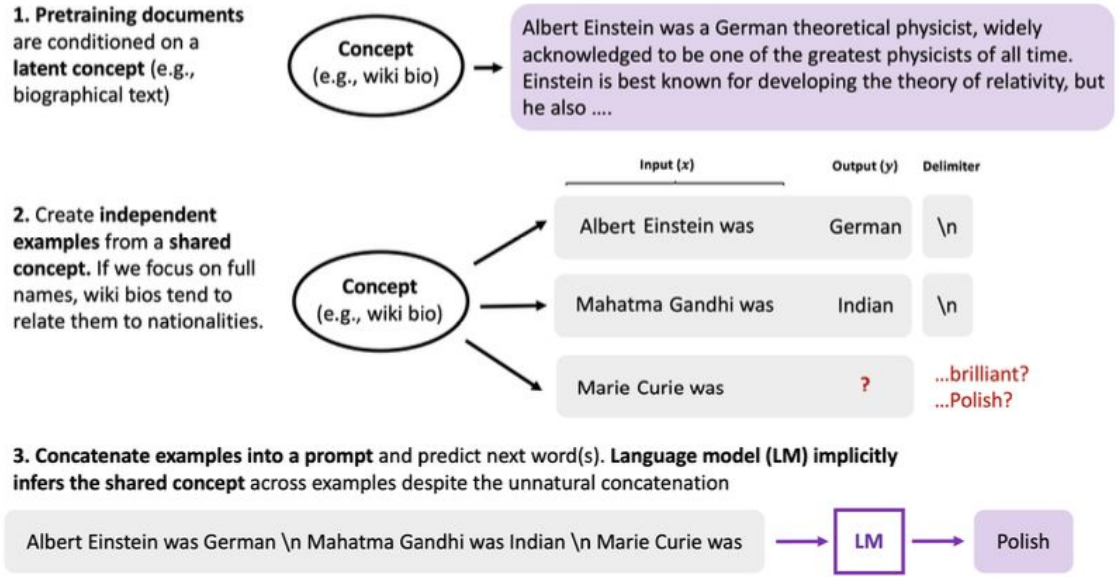
$$p(\text{output}|\text{prompt}) = \int_{\text{concept}} p(\text{output}|\text{concept}, \text{prompt}) p(\text{concept}|\text{prompt}) d(\text{concept})$$

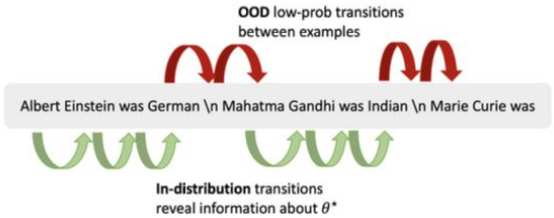
- Prompt provides evidence for model to sharpen the posterior distribution over concepts
- *p(concept | prompt)* concentrates on the underlying prompt concept
- We want *p(concept | prompt)* to converge to a delta distribution and pick out the correct concept

Key logical leap:

- LM will infer **prompt concept** from in-context examples, even though prompts are sampled from a very different distribution!
- Connection to pre-training: to generate coherent text over time, it must learn underlying concept

Example of a prompt

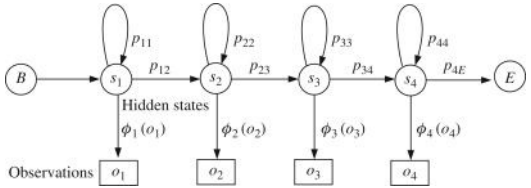




Pre-training distribution

- Each document is a length T sequence sampled by:

$$p(o_1, \ldots, o_T) = \int_{\theta \in \Theta} p(o_1, \ldots, o_T|\theta) p(\theta) d\theta$$



- $p(o_1, o_2, .., o_T \mid \theta)$ defined by Hidden Markov Model
- θ defines transition probability matrix for hidden states $h_1, \ldots, h_T$
- Intuitively, θ models document-level statistics such as format, sentiment, topic, etc.
- We wish to infer θ from the prompts
- Note: This is an **assumption** about how text is generated
- Assumption: Language model and data **large** enough to fit pre-training distribution
  i.e. $p_{model} = p_{text} = p$

Prompt distribution

- For i = 1, …, n, the i^{th} demonstration $O_i = [x_i, y_i]$ where x_i is input token sequence, y_i is output token
- Each O_i independently generated using:
  1. Generate start hidden state h_i^{start} from prompt start distribution p_{prompt}(ideally included in θ? Why the same distribution?)
  2. $p(Oi \mid h_i^{start}, \theta^*)$ = **Pre-training** distribution conditioned on concept $\boldsymbol{\theta}^*$

- Prompt is a sequence of demonstrations S_n followed by test example x_{test}:

$$[S_n, x_{test}] = [O_1, o^{delim}, O_2, o^{delim}, \ldots, o^{delim}, O_n, x_{test}] = [x_1, y_1, o^{delim}, x_2, y_2, o^{delim}, \ldots, x_n, y_n, o^{delim}, x_{test}] \sim p_{prompt}$$

Key Result

By structure of prompt: $$y_{\text{test}} \sim p_{\text{prompt}}(y|x_{\text{test}}) = \mathbb{E}_{h_{\text{test}}^{\text{start}} \sim p_{\text{prompt}}(h_{\text{test}}^{\text{start}}|x_{\text{test}})} \left[p(y|x_{\text{test}}, h_{\text{test}}^{\text{start}}, \theta^*) \right]$$

Abuse of notation: doesn't p_{prompt} capture sequence of prompts? How can it capture $p_{prompt}(h_{test}^{start} | x_{test})$ or $p_{prompt}(y|x_{test})$?

Under some assumptions, as n → ∞,

$$argmax_y p(y|S_n, x_{test}) \rightarrow argmax_y p(y|\theta^*, x_{test}) \rightarrow argmax_y p_{prompt}(y|x_{test})$$

What we can obtain from LM

What we show

By definition of prompt

More examples → More signals for Bayesian Inference → Smaller Error

Heuristic Derivation

$$p(y|S_n, x_{\text{test}}) = \int_\theta p(y|S_n, x_{\text{test}}, \theta) p(\theta|S_n, x_{\text{test}}) d\theta$$

$$\propto \int_\theta p(y|S_n, x_{\text{test}}, \theta) p(S_n, x_{\text{test}}|\theta) p(\theta) d\theta \quad \text{(Bayes' rule, drop the constant } \frac{1}{p(S_n, x_{\text{test}})}\text{)}$$

$$= \int_\theta \sum_{h_{\text{test}}^{\text{start}} \in \mathcal{H}} p(y|x_{\text{test}}, h_{\text{test}}^{\text{start}}, \theta) p(h_{\text{test}}^{\text{start}}|S_n, x_{\text{test}}, \theta) \frac{p(S_n, x_{\text{test}}|\theta)}{p(S_n, x_{\text{test}}|\theta^*)} p(\theta) d\theta$$

(Law of total prob, Markov property, divide by $p(S_n, x_{\text{test}}|\theta^*)$ (a constant))

This is where we get rid of S_n using Markov property and are now left with θ^*

$$= \int_\theta \sum_{h_{\text{test}}^{\text{start}} \in \mathcal{H}} p(y|x_{\text{test}}, h_{\text{test}}^{\text{start}}, \theta) p(h_{\text{test}}^{\text{start}}|S_n, x_{\text{test}}, \theta) \exp(n \cdot r_n(\theta)) p(\theta) d\theta$$

$$exp(n.r_n(\theta)) \to 0 \quad \forall \quad \theta \neq \theta^*$$
$$exp(n.r_n(\theta)) \to 1 \quad \forall \quad \theta = \theta^*$$

→ Simplify → $p(y \mid x_{test}, \theta^*)$

Sketch for limit of $r_n(\theta)$

Key challenge: Sequence of examples S_n in *p(.)* while *p(.)* generates each example independently

Solution: Factorise examples using assumptions on delimiter tokens i.e. prove:

$$p(S_n, x_{\text{test}}|\theta) = p(x_{\text{test}}|S_n, \theta)p(S_n|\theta) \approx \prod_{i=1}^{n} O(1)p(O_i|\theta)$$

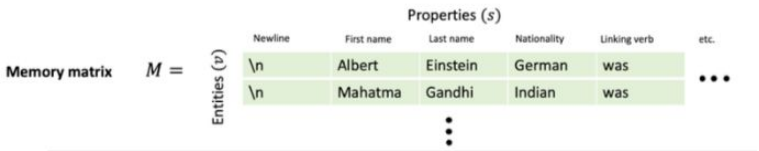
Then, we can upper bound $r_n(\theta)$: $r_n(\theta) \leq \frac{1}{n}\left(O(n) + \sum_{i=1}^{n} \log \frac{p(O_i|\theta)}{p(O_i|\theta^*)}\right) \to O(1) + \mathbb{E}_{O\sim p_{\text{prompt}}}\left[\log \frac{p(O|\theta)}{p(O|\theta^*)}\right]$

$$KL(p_{\text{prompt}}(O)\|p(O|\theta^*)) - KL(p_{\text{prompt}}(O)\|p(O|\theta))$$

< c

Break into k (# tokens) KL terms
These are large due to mismatch

Generative IN-Context learning (GINC) dataset

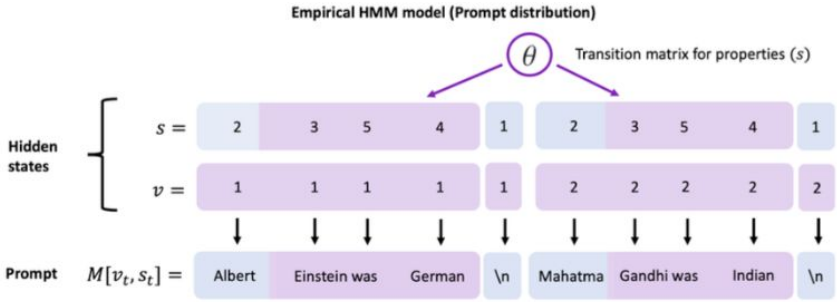


- HMM hidden state at time t:
  $h_t = [v_t, s_t]$ where
  v_t = entity (e.g. Einstein)
  s_t = property (e.g. nationality, last name, etc.)
- Entities and Properties are modelled as **independent** Markov chains
- Emission token is deterministic
  given v_t and s_t: $M[v_t, s_t]$
- Entity changes slowly: p(change) < 0.1
- Property changes quickly

GINC (continued)

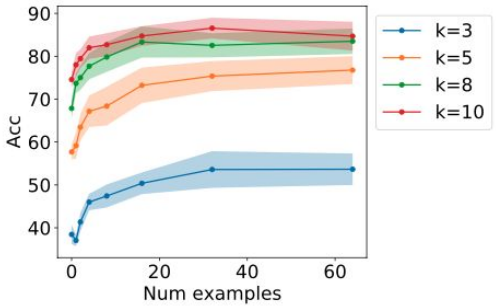
- Concept θ: property transition matrix
  5 Transition matrices generated independently (one for each concept)
  E.g. one for wiki bios, one for conversation, one for news
- Entity transition fixed for all concepts (Why? Leads to different formatting of same set of ideas)
- Uniform mixture of HMMs over 5 concepts generates 1000 documents with ~10 million tokens total
- Start distribution: **uniform** across all 100 hidden states

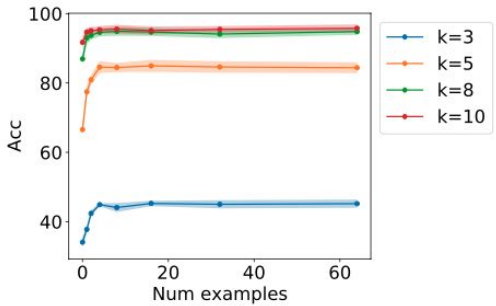
GINC prompt generation



1. Sample concept θ uniformly at random (choosing HMM mixture)
2. Then, sample uniformly for entities but fixed starting property (sampled uniformly) to maintain consistency in task e.g.
    a. Entity: Sample uniformly from {Curie, Gandhi, Curie} for each example
    b. Property: Sample uniformly from {Name, DOB month, Nationality} and **fix** for this prompt
    c. Generate *k* tokens using HMM
    d. Repeat a and c while using fixed start property from b
3. For the last example, generate *k - 1* tokens and use last as ground truth

Simulations





K = length of each example

Accuracy = Number of correct output predictions

[Chance perf. = 1/vocab size where vocab size in {50, 100, 150}]

Accuracy ↑ with

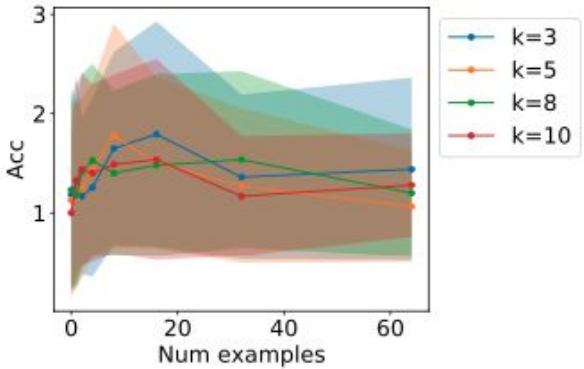
- ↑ sequence length
- ↑ number of examples

Intuition: both help distinguish between transition matrix of concepts

Expected as more signal for inferring concept

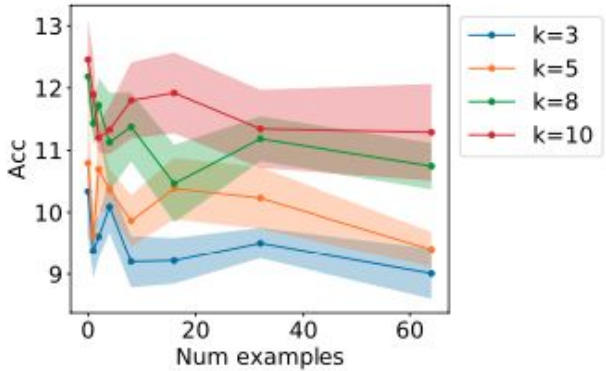
LSTM >~ Transformer!

Empirical evidence for inferring θ*



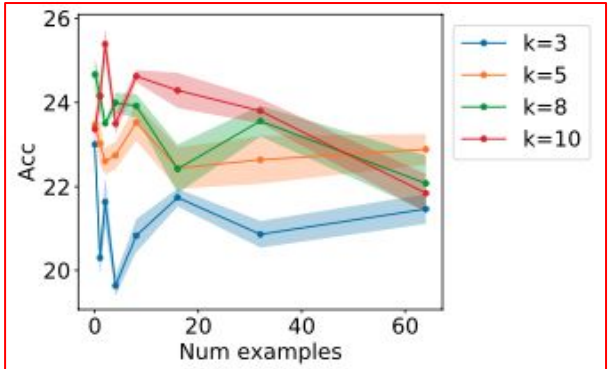
Pre-train on random transitions -> chance perf.

So? No underlying structure



Unseen concepts -> fails to extrapolate

So? Helps test if a concept was present in PT data. Consequence: If Wiki bio never seen during PT, difficult to expect the model to complete nationalities in the given format



Pre-trained on only one concept

Flat curves

When there is no underlying concept

No explanation for experiment with only one concept!!

Guesses:

- Intuitively, if we have only concept, the task is **easier**?
- If only one concept, say Wiki bios, transition matrix (for properties) is fixed
- Model, during PT, still needs to learn the **transitions** for next-token prediction
- What is it instead learning, in order to minimize PT loss?
- Does diversity in concepts force it to **factorise text into properties and entities**, which are crucial for in-context learning? But model is very large? No forced factorization?

Effect of model size and architecture

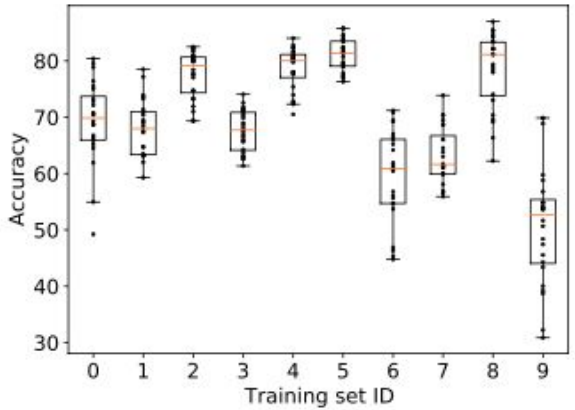
Model	# Params	Train loss (pretraining)	Val loss (pretraining)	In-context Acc
Vocab size 50, $k = 10, n = 64$				
Transformer (4 layer)	29M	1.49	1.50	60.2 ± 5.7
Transformer (12 layer)	85M	1.31	1.33	81.2 ± 7.1
Transformer (16 layer)	115M	1.31	1.33	84.7 ± 3.4
LSTM	28M	1.31	1.35	95.8 ± 1.11
Vocab size 100, $k = 10, n = 64$				
Transformer (4 layer)	29M	1.58	1.59	67.4 ± 4.7
Transformer (12 layer)	85M	1.40	1.42	84.6 ± 3.0
Transformer (16 layer)	115M	1.41	1.43	88.7 ± 1.6
LSTM	28M	1.43	1.44	95.8 ± 1.54
Vocab size 150, $k = 10, n = 64$				
Transformer (4 layer)	29M	1.44	1.45	92.8 ± 1.9
Transformer (12 layer)	85M	1.27	1.28	98.4 ± 0.4
Transformer (16 layer)	115M	1.27	1.28	98.1 ± 0.5
LSTM	28M	1.26	1.31	99.2 ± 1.06

Observation:
Even when pre-train loss is same, performance increases

Explanation:
Overparameterization

LSTM > Transformer
Why? Similar to HMM in architecture!

Sensitivity to Example Ordering



Prompts generated from single context

Each training set ID contains 4 prompts

Each of the 4! = 24 permutations of these 4 prompts is one single dot

10-40% variation in performance! (reported in another paper)

Not good!!

Questions

- Prompts such as [“News” // positive] is also low probability?
- Where in the architecture is the concept found? Is it distributed across weights or can we extract is from one of the layers?
- Where is $M[v_t, s_t]$ stored in the model?

References

Original paper: *Xie, Sang Michael, et al. "An explanation of in-context learning as implicit bayesian inference." arXiv preprint arXiv:2111.02080 (2021).*

Blog Post by authors: How does in-context learning work? A framework for understanding the differences from traditional supervised learning | SAIL Blog