Beyond the Black Box: A Statistical Model for LLM Reasoning and Inference

Shin Yun Seop

August 6, 2025

Seoul national university, statistics, IDEA LAB

Contents

Motivation

- Since ChatGPT came out, large language models have drawn a lot of attention. Many studies now ask why they can handle tasks like in-context learning.
- This paper uses a Bayesian model to explain these behaviors.
- Bayesian models are natural here since tokens are being generated based on the past training data (prior) and the prompts (new observations which updates the prior).

Contents

Ideal Generative Text Model

- Probability matrix: each prompt is a row, each token is a column. The cell holds the probability the token comes next.
- Generation is simple. Find the row for the current prompt, sample one token from its multinomial, append it, then jump to the row for the new prompt.
- Example: With prompt "Protein", high-probability tokens include "synthesis"and "shake". Choosing "synthesis" moves us into a biology-centric prompt, while "shake" shifts the model toward fitness-drink prompt.

Ideal Generative Text Model

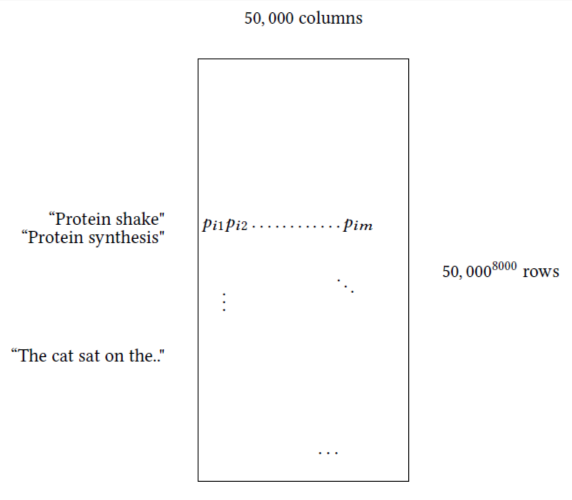

Figure 1: Example of ideal multinomial probability matrix

Real-World LLM

- The ideal probability matrix is very large. Far too big to store, and get its element is almost impossible.
- LLMs handle this by compressing the huge matrix into learned weights: a prompt $\rightarrow$ embedding $\rightarrow$ probability which is a parameter of multinomial distribution.
- It works fine for prompts that look like the training data, but for unfamiliar prompts the model may give odd probabilities.

Ideal model vs Real-World Models

	Ideal probability matrix	**Real-world LLM**
Storage	Stores every prompt as its row ("The catis" vs "The cat is" live in different rows).	Keeps no full rows. Prompt $\rightarrow$ embedding e, then $p = \mathrm{softmax}(We + b)$.
Outcome	Rows are independent, so probability vectors p_1, p_2 can differ arbitrarily.	One-token or whitespace change $\Rightarrow e_1 \approx e_2 \Rightarrow p_1 \approx p_2$.
Property	Assumes infinite memory	

Contents

Notation

- v : Total number of token (In this paper assume $v = 50,000$).
- $\mathcal{E}$: Embedding space (In this paper assume $\mathcal{E} = \mathbb{R}^r$ for some r).
- $\mathcal{P} \subset \mathbb{R}^v$: Space of probability vector such that if $p = (p_1, \cdots, p_v) \in \mathcal{P}$ then $p_j \geq 0$ for all $j = 1, \cdots, v$ and $p_1 + \cdots + p_v = 1$.
- $T : \mathcal{E} \to \mathcal{P}$: Convexity preserving mapping (In this case decoder part of LLM).

Continuity of Embedding Mapping

Theorem 1 (Continuity)

If the mapping $T : e \mapsto p(e)$ from a prompt embedding $e \in \mathcal{E}$ to its next-token probability $p(e) \in \mathcal{P}$ is *convexity preserving* and *bounded*, then T is **continuous**.

- This means that small changes in e cause only small changes in $p(e)$.
- This continuity property lets the compressed model generalize beyond seen prompts and forms the basis for the Bayesian update.

Dirichlet Mixture Prior

Theorem 2 (Dirichlet Mixture Approximation)

Any continuous bounded prior $u(P)$ over $\mathcal{P}$ can be approximated by a **finite mixture of Dirichlet distributions**:

$$u(P) \approx \sum_{k=1}^{n} b_k \, \mathrm{Dir}(P \mid \boldsymbol{\alpha}_k), \qquad b_k \geq 0, \sum b_k = 1.$$

- It is the conjugate prior for the multinomial likelihood, so posterior updates are easy and have a closed form solution.
- This means mixing a few Dirichlet components can mimic any shape, from very flexible distribution to almost uniform distribution.
- After seeing new tokens, you just add their counts; the result is still a Dirichlet mixture, so updating stays easy and fast.

Contents

LLM as Bayesian Learning

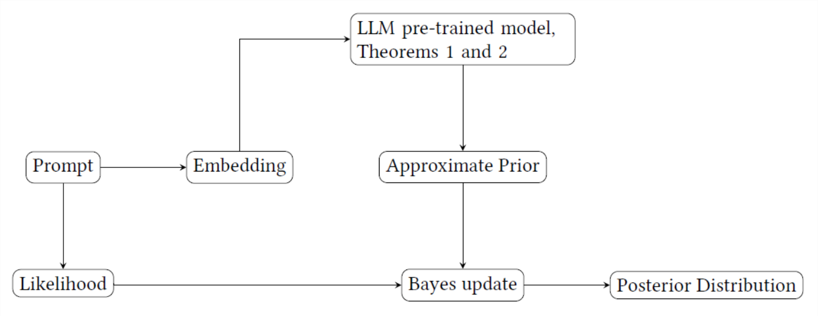

Figure 2: Bayesian updating of next token multinomial probability.

LLM as Bayesian Learning

- **Prior.** Prompt embeddings correspond to an approximation of a Dirichlet–mixture prior $u(P) = \sum_k b_k \, \mathrm{Dir}(P \mid \boldsymbol{\alpha}_k)$.
- **Likelihood.** n tokens inside the prompt give counts $\mathbf{c} = (c_1, \ldots, c_V)$.
- **Posterior update.** By Conjugate property $\boldsymbol{\alpha}_k \leftarrow \boldsymbol{\alpha}_k + \mathbf{c}$.

In-Context-Learning in Example

- Now think simple example of explain how do In-Context learning in LLM.
- For simplicity we think binary target case.
- Example1 : I like Jisu, because she is bad girl.
- Example2 : I like Minsu, because he is bad boy.
- Example2 : I like Minho, because he is bad person.
- Q: I like Juho, because he is $\{\quad\}$. (Target: nice, bad)

In-Context-Learning in Example

- In this case we can model

$$\text{nice} \mid p \sim \text{B}(3,p) \text{ and } p \sim \text{Beta}(\alpha, \beta).$$

	Prior (α, β)	**Observed** (nice, bad)	**Posterior mean**
Strong prior	$(5, 1)$	$(0, 3)$	$5/(5+1+3) \approx 0.56$
Weak prior	$(0.3, 0.01)$	$(0, 3)$	$0.3/(0.3+0.01+3) \approx 0.09$

$$E[\text{nice} \mid \text{Observed}] = \frac{\alpha}{\alpha+\beta+n}, \quad E[\text{bad} \mid \text{Observed}] = \frac{\beta+n}{\alpha+\beta+n}.$$

- Weak prior: even $n = 3$ flipped observation, the posterior is varying with prior.
- Strong prior: requires many observed prompts and slow adaptation.
- Authors argue that larger model tend to have many more parameters and during training they are acquiring more general knowledge so this results in small $\alpha + \beta$ (Why??).

Summary

- This paper formalizes how real-world LLMs work and proves a continuity theorem, showing that prompt embeddings map smoothly to multinomial token distributions. I.e., even a slight change in the prompt should not cause a sudden shift in the predicted token distribution.
- Bayesian Explanation: Rethink next-token prediction as a Bayesian posterior (Dirichlet-mixture prior + prompt likelihood).

Appendix: Llama example

Natural Language Query	DSL representation
Tournament0 team with best win loss record after losing the toss	{'orderby': ['win_loss_ratio'], 'toss': ['lost'], 'tournament': ['Tournament0'], 'type': ['team']}
lowest team total	{'groupby': ['innings'], 'orderby': ['runs'], 'sortorder': ['reverse'], 'type': ['team']}
biggest Tournament0 total in defeat	{'groupby': ['innings'], 'orderby': ['runs'], 'result': ['loss'], 'tournament': ['Tournament0'], 'type': ['team']}
highest scores by Team0	{'groupby': ['innings'], 'orderby': ['runs'], 'team': ['Team0'], 'type': ['team']}

Figure 3: Few shot examples of NL $\rightarrow$ DSL(Domain Specific Language)

Appendix: Llama example

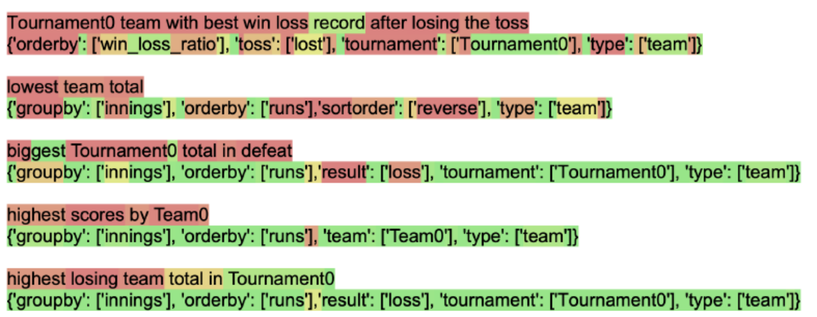

Figure 4: Probability: Red $\leq$ Yellow $\leq$ Green. The first four are the few-shot examples, and the last one is our query.