Uncertainty Quantification for LLM

Presenter: YunSeop Shin

May 20, 2026

Seoul national university, statistics, IDEA LAB

Contents

Decomposing Uncertainty for Large Language Models through Input Clarification Ensembling (ICML 2024)

Introduction

- This paper proposes input clarification ensembling to quantify predictive uncertainty in LLMs by generating multiple clarifications of an ambiguous input and ensembling the resulting predictions.
- This method is mainly designed to estimate aleatoric uncertainty induced by ambiguity of text input.
- Inspired by Bayesian neural networks and deep ensembles, this framework replaces parameter space ensembling with input space, since modifying or ensembling LLM parameters is often impractical.

Notation

- X : Text input.
- $Y \in \mathbb{R}^c$: Output target.
- θ: Parameters of an LLM (In this paper, θ is constant).
- $q(Y|X, \theta)$: Predictive distribution of Y given X. For brevity, the authors often write $q(Y|X)$, since the model parameters θ are treated as fixed.
- C : A clarification of the input, i.e., an additional text of an ambiguous prompt that makes its meaning more explicit.
- $p(C|X)$:The distribution of the clarification given a X.
- $\mathcal{H}(p) = -\sum_{j=1}^{c} p_j \log p_j$ for a probability vector $p \in \mathbb{R}^c$. Entropy, used as a measure of uncertainty in the predictive distribution.
- $\oplus$: concatenation.

Example

- X : 'Who is the president of this country?'
- C : 'This country refers to the US.'
- This method generates many clarifications $C^{(k)}$ using LLM. And then ensemble the prediction of $X \oplus C^{(k)}$.

Uncertainty Decomposition

- Authors define the predictive distribution $q(Y|X)$ as an ensemble of predictions conditional on each clarified input:

$$q(Y|X) = \mathbb{E}_{p(C|X)}[q(Y|X \oplus C)] \tag{1}$$

- Then, propose to decompose the uncertainty of the ensemble model as:

$$\mathcal{H}(q(Y|X)) = \mathcal{I}(Y; C|X) + \mathbb{E}_{p(C|X)}\mathcal{H}(q(Y|X \oplus C)). \tag{2}$$

- Note that if X is clear then C is almost empty, then $\mathcal{I}(Y; C|X) \approx 0$.
- In contrast, when the input is highly ambiguous, the resulting clarifications can differ substantially, implying that $\mathcal{I}(Y; C \mid X)$ is nonzero.

Uncertainty Decomposition

- Based on this intuition, $\mathcal{I}(Y; C \mid X)$ serves as an estimate of the aleatoric uncertainty caused by input ambiguity.
- In contrast, $\mathbb{E}_{p(C|X)}\mathcal{H}(q(Y \mid X \oplus C))$ measures the average uncertainty on clarified inputs, so the remaining uncertainty can mostly be ascribed to epistemic uncertainty.
- Compared with Bayesian neural networks and deep ensembles, the corresponding roles of aleatoric and epistemic uncertainty are swapped.

Appendix

- How to sample $C^{(k)}$ from $p(C \mid X)$?
- In-context learning: prompt a clarification LLM, such as GPT-3.5-Turbo or GPT-4, with task instructions and several in-context examples.
- Supervised fine-tuning: fine-tune Llama-3-8B-Instruct on datasets of ambiguous inputs and their corresponding clarifications.

Appendix

- To estimate $q(Y \mid X \oplus C)$, the clarified input $X \oplus C$ is fed to the prediction LLM and 10 answers are sampled.
- Since the task is open-ended generation, semantically equivalent answers are clustered into the same group.
- The empirical frequencies of these answer groups are then normalized to form an estimate of $q(Y \mid X \oplus C)$.

Fine-Grained Uncertainty Decomposition in Large Language Models: A Spectral Approach (NeurIPS 2025 Workshop)

Introduction

- A follow up research of input clarification ensembling for uncertainty decomposition in LLMs.
- The main limitation of the previous approach is that its clustering-based probability provides only a coarse semantic representation.
- This paper introduces a decomposition using kernel-based von Neumann entropy.

Kernel-based von Neumann Entropy

- Z: an arbitrary random variable taking values in $\mathcal{Z}$, and P_Z its distribution.
- $k : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$: a given positive semidefinite kernel, and $\mathcal{H}$ the corresponding RKHS.
- $\Phi(z) = k(\cdot, z) \in \mathcal{H}$: the corresponding feature map.
- The RKHS satisfies the reproducing property $f(z) = \langle f, \Phi(z) \rangle_{\mathcal{H}}$ for all $f \in \mathcal{H}$.
- For $f, g, h \in \mathcal{H}$, the tensor product operator $(f \otimes_{\mathcal{H}} g) : \mathcal{H} \to \mathcal{H}$ is defined by

$$(f \otimes_{\mathcal{H}} g)(h) = \langle g, h \rangle_{\mathcal{H}} f \tag{3}$$

Kernel-based von Neumann Entropy

- The non-central covariance operator associated with P_Z is defined as

$$\Sigma_{P_Z} := \mathbb{E}_{Z \sim P_Z}\left[\Phi(Z) \otimes_{\mathcal{H}} \Phi(Z)\right]. \tag{4}$$

- Let $\lambda_1(\Sigma_{P_Z}), \lambda_2(\Sigma_{P_Z}), \ldots$ denote the eigenvalues of Σ_{P_Z}.
- The kernel-based von Neumann entropy is defined by

$$H_{VN}(P_Z) := -\operatorname{Tr}\left[\Sigma_{P_Z} \log \Sigma_{P_Z}\right] = -\sum_{i=1}^{\infty} \lambda_i(\Sigma_{P_Z}) \log \lambda_i(\Sigma_{P_Z}). \tag{5}$$

Kernel-based von Neumann Entropy

- In practice, $H_{VN}(P_Z)$ is estimated empirically from sampled observations $Z_1, Z_2, \cdots, Z_n \sim P_Z$ via the empirical covariance operator

$$\hat{\Sigma}_{P_Z} = \frac{1}{n} \sum_{i=1}^{n} \Phi(Z_i) \otimes_{\mathcal{H}} \Phi(Z_i). \tag{6}$$

- This yields the plug-in estimator

$$\hat{H}_{VN}(P_Z) = - \sum_{\hat{\lambda} \in \Lambda(\hat{\Sigma}_{P_Z})} \hat{\lambda} \log \hat{\lambda}. \tag{7}$$

- Since the nonzero eigenvalues of $\hat{\Sigma}_{P_Z}$ coincide with the eigenvalues of the Gram matrix $\frac{1}{n}K$, the estimator can be computed as

$$\hat{H}_{VN}(P_Z) = - \sum_{i=1}^{n} \hat{\lambda}_i \log \hat{\lambda}_i, \tag{8}$$

where $\hat{\lambda}_i$ denotes the eigenvalues of $\frac{1}{n}K$ s.t. $K_{ij} = k(Z_i, Z_j)$.

Notation

- X: text input.
- C: a clarification of the input.
- $p(C|X)$:The distribution of the clarification given a X.
- A: text answer generated by the LLM given $X \oplus C$.
- f_{emb}: a given embedding function.
- $Y = f_{emb}(A)$: embedding of a generated answer.
- $Q_{Y|X\oplus C}, Q_{Y|X}$: conditional and marginal distributions of Y w.r.t. C in the embedding space.
- $\mathcal{H}_{VN}(P)$: kernel-based von Neumann entropy of a given distribution P.
- $\oplus$: concatenation.

Uncertainty Decomposition

- As in the previous clarification ensembling, this work condition on the clarification variable C.
- However, this paper does not use a mutual information based decomposition.
- Instead, by applying the general decomposition for concave uncertainty measures to the kernel-based von Neumann entropy, obtain

$$\mathcal{H}_{VN}(Q_{Y|X}) = \mathbb{E}_{p(C|X)}\left[\mathcal{H}_{VN}(Q_{Y|X\oplus C})\right] + \mathcal{B}_{-\mathcal{H}_{VN}}(Q_{Y|X\oplus C}). \tag{9}$$

- Here, the first term is interpreted as epistemic uncertainty, while the second term is the VN entropy induced functional Bregman information, which plays the role of aleatoric uncertainty.

Estimating the Decomposition

- Sample $C^{(1)}, \cdots, C^{(n)} \sim p(C|X)$.
- For each clarification $C^{(i)}$, generate m answers $A_{i1}, \cdots, A_{im}$ from the target LLM, and obtain their embeddings $Y_{ij} = f_{emb}(A_{ij})$.
- For each fixed $C^{(i)}$, the set $\{Y_{ij}\}_{j=1}^{m}$ is considered as samples of $Q_{Y|X\oplus C^{(i)}}$ so, used to estimating the $\mathcal{H}_{VN}(Q_{Y|X\oplus C^{(i)}})$.
- Therefore,

$$\mathbb{E}_{p(C|X)}\left[\mathcal{H}_{VN}(Q_{Y|X\oplus C})\right] \approx \frac{1}{n}\sum_{i=1}^{n}\mathcal{H}_{VN}(Q_{Y|X\oplus C^{(i)}}). \quad (10)$$

- Pooling all samples $\{Y_{ij}\}_{i=1,\cdots,n,\, j=1,\cdots,m}$ are considered as samples from the marginal distribution $Q_{Y|X}$, which is used to estimate $\mathcal{H}_{VN}(Q_{Y|X})$.

Textual Bayes: Quantifying Uncertainty in LLM-Based Systems (ICLR 2026)

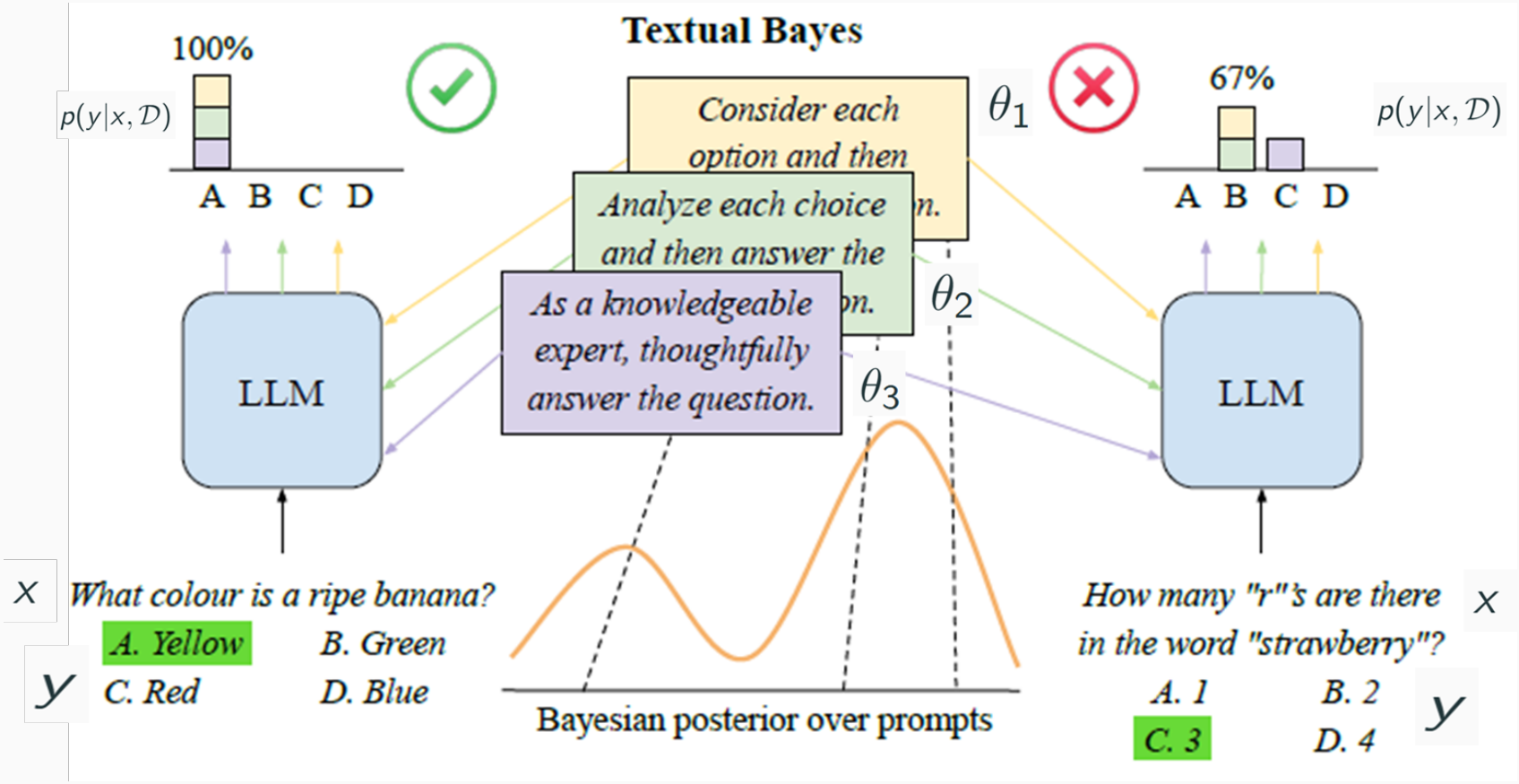
Introduction

- The first two papers concentrate on the question X, whereas this paper centers on the prompt.
- Interpreting prompts as parameters of a statistical model enables Bayesian inference over prompts using a training dataset.
- Authors introduce Metropolis-Hastings through LLM Proposals (MHLP), a Markov chain Monte Carlo (MCMC) algorithm that combines prompt optimization techniques (TextGrad) with standard MCMC methods (MH).

Notation

- X : Text input (e.g. a question)
- Y : Output (e.g. the model's predicted answer)
- θ : a prompt (e.g. a system message defining instructions for the LLM's behavior)
- $\mathcal{D} = \{(x_1, y_1), \cdots, (x_n, y_n)\}$: Given an i.i.d. dataset.
- $\hat{Y} = \textbf{LLM}(X; \theta)$: output generated by the LLM for input X under prompt parameter θ.
- $p(Y \mid X, \theta)$: LLM induced distribution over outputs y given input x and prompt θ.

Overall Structure



- The problem is how to sample $\theta_1, \cdots, \theta_m$ from $\pi(\theta \mid \mathcal{D})$.
- This paper solves this problem using Metropolis-Hastings MCMC.

Model

Prior When $\theta = (\text{Token}_1, \cdots, \text{Token}_T)$ and prior_meta_prompt be are given, $\pi(\theta)$ is following

$$\pi(\theta) = \prod_{t=1}^{T} p_{\mathbf{LLM}}(\text{Token}_t | \text{prior\_meta\_prompt}, \text{Token}_{<t}). \tag{11}$$

Example of prior_meta_prompt: "Generate a good system prompt for a fact-seeking question-answering language model. The system prompt must be concise and generic and work with any question."

Posterior $\pi(\theta|\mathcal{D})$ is formally defined using prior and likelihood

$$\pi(\theta|\mathcal{D}) \propto \pi(\theta) \prod_{i=1}^{n} p(y_i|x_i, \theta) = g(\theta). \tag{12}$$

Proposal distribution

- Given θ, TextGrad proposes θ', thereby defining $q(\theta' \mid \theta)$.
- The proposed prompt is then accepted or rejected via stochastic Metropolis-Hastings. The acceptance probability is given by

$$\begin{aligned}\log \alpha = &\log \pi(\theta') - \log \pi(\theta) + \beta \left\{\log p(y_i|x_i, \theta') - \log p(y_i|x_i, \theta)\right\} \\ &+ \log q(\theta|\theta') - \log q(\theta'|\theta)\end{aligned} \tag{13}$$

  where β is given hyperparameter.

- Here, initial θ (i.e. $\theta^{(0)}$) is "Answer the question. Think step-by-step. "

Appendix

- prior_loss(θ) = **LLM**(prior_losses_text,θ).

Evaluate if the system prompt is concise and generic.
Provide feedback on how to improve it.
Prompt
Answer the question based on your knowledge. Think step-by-step.
Output
The system prompt is reasonably concise but could be more specific about verification and accuracy. Consider adding instructions about cross-referencing sources and focusing on the core question. The prompt should also emphasize conciseness more explicitly.

Appendix

- likelihood_loss$(X, Y, \hat{Y}) =$
  LLM(evaluation_instruction$, X, Y, \hat{Y}$).

You will be given a question and an answer attempted by a language model. Evaluate the attempted answer. Be smart, logical, and very critical. Do not solve the question. Just provide concise feedback.
Question What was the winner of 1993 Scandinavian Masters?
Correct Answer Per-Ulrik Johansson
LLM Answer The winner of the 1993 Scandinavian Masters golf tournament was Per-Ulrik Johansson. He secured the title held in Sweden, which was part of the European Tour. Answer: Per-Ulrik Johansson

Appendix

- total_loss = **Concat**(likelihood_loss(X, Y, $\hat{Y}$), prior_loss(θ))
- Obtain total_loss.backward using TextGrad.
- $\theta^{'}$ = **LLM**(θ, total_loss.backward)
- When $\theta' = (\text{Token}'_1, \cdots, \text{Token}'_{T'})$

$$q(\theta'|\theta) = \prod_{t=1}^{T'} p_{\textbf{LLM}}(\text{Token}'_t|\theta, \text{total\_loss.backward}, \text{Token}'_{<t}). \tag{14}$$

LUQ: Long-text Uncertainty Quantification for LLMs (ACL 2024)

Introduction

- This paper proposes a method for quantifying uncertainty in long responses generated by large language models (LLMs).
- The key idea is to measure response consistency at the sentence level, or at a finer-grained atomic level, and to use these consistency scores to estimate the overall uncertainty of a response.
- For a given question, the method first generates multiple sampled responses from the LLM.
- It then measures sentence-level support across the sampled responses, aggregates these scores into response-level support, and finally computes an uncertainty score for the input question x.

Notation

- x: a given query.
- $R = \{r_1, r_2, \cdots, r_m\}$: the set of m sampled responses generated for the same query x.
- s_{ij}: the j-th sentence in response r_i $(j = 1, \cdots, L_i)$.
- For $i \neq k$, $\ell_e(s_{ij}, r_k)$ and $\ell_c(s_{ij}, r_k)$ denote the entailment logit and the contradiction logit, respectively, produced by the NLI (Natural Language Inference) classifier when the sentence s_{ij} is compared against the response r_k.

Sentence-level support score P

For each sentence s_{ij} in response r_i, we measure how strongly it is supported by another sampled response r_k.

$$P_{ijk} := P(\text{entail} \mid s_{ij}, r_k) = \frac{\exp\big(\ell_e(s_{ij}, r_k)\big)}{\exp\big(\ell_e(s_{ij}, r_k)\big) + \exp\big(\ell_c(s_{ij}, r_k)\big)}, \quad i \neq k.$$

- P_{ijk} is the sentence-level support score.
- A larger P_{ijk} means that the sentence s_{ij} is more strongly supported by the response r_k.

Response-pair support score S

We aggregate the sentence-level support scores over all sentences in r_i to obtain the support of response r_i from response r_k.

$$S_{ik} := \frac{1}{L_i} \sum_{j=1}^{L_i} P_{ijk}, \quad i \neq k.$$

- S_{ik} is the response-level support score from r_k to r_i.
- A larger S_{ik} indicates that the response r_i is more consistent with r_k.

Response-level confidence score C

We next average the pairwise support scores over all other sampled responses.

$$C_i := \frac{1}{m-1} \sum_{\substack{k=1 \\ k \neq i}}^{m} S_{ik}, \quad i = 1, \ldots, m.$$

- C_i is the confidence score of response r_i.
- A larger C_i means that r_i is more consistently supported by the other sampled responses.

Query-level uncertainty score U

Finally, we average the lack of confidence over all sampled responses to obtain the uncertainty for the query x.

$$U(x) := \frac{1}{m} \sum_{i=1}^{m} (1 - C_i).$$

- $U(x)$ is the overall uncertainty score for the query x.
- A smaller $U(x)$ indicates that the sampled responses are more mutually consistent.
- A larger $U(x)$ indicates greater disagreement across the sampled responses, and hence higher uncertainty.

변형

$$U(x) = 1 - \frac{1}{m}\sum_{i=1}^{m} C_i = 1 - \frac{1}{m}\frac{1}{n}\sum_{i=1}^{m}\sum_{\substack{k=1\\k\neq i}}^{m} S_{ik} = 1 - \frac{1}{m}\frac{1}{n}\sum_{i=1}^{m}\frac{1}{L_i}\sum_{\substack{k=1\\k\neq i}}^{m}\sum_{j=1}^{L_i} P_{ijk}$$

$$= 1 - \frac{1}{m}\sum_{i=1}^{m}\frac{1}{L_i}\sum_{j=1}^{L_i}\frac{1}{n}\sum_{\substack{k=1\\k\neq i}}^{m} P_{ijk} = 1 - \frac{1}{m}\sum_{i=1}^{m}\frac{1}{L_i}\sum_{j=1}^{L_i} M_{ij}$$

$$= \frac{1}{m}\sum_{i=1}^{m}\frac{1}{L_i}\sum_{j=1}^{L_i}(1 - M_{ij}) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{L_i}\sum_{j=1}^{L_i} u_{ij}$$

- $u_{ij} = 1 - \frac{1}{n}\sum_{\substack{k=1\\k\neq i}}^{m} P_{ijk}$ 는 i번째 응답의 j번째 문장의 불확실성으로 볼 수 있을까??
- 이를 확장해서 일반적인 응답의 j번째 문장의 불확실성을 구할수 있을까?
- 그러기 위해서는 response를 꼭 추가로 n개 만들 수 밖에 없을까?

Reference

- Hou, Bairu, et al. "Decomposing Uncertainty for Large Language Models through Input Clarification Ensembling." International Conference on Machine Learning. 2024.
- Walha, Nassim, et al. "Fine-Grained Uncertainty Decomposition in Large Language Models: A Spectral Approach." NeurIPS 2025 Workshop: Reliable ML from Unreliable Data.
- Ross, Brendan Leigh, et al. "Textual Bayes: Quantifying uncertainty in LLM-based systems." arXiv preprint arXiv:2506.10060 (2025).