SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

November 2025, Vol. 68, Iss. 11, 210101:1–210101:20
https://doi.org/10.1007/s11432-024-4550-8

From CAS & CAE Members

Uncertainty-aware large language model response length perception

Bin SHI[1,2], Bo DONG[2,3] & Qinghua ZHENG[1,2*]

[1]*School of Computer Science and Technology, Xi'an Jiaotong University, Xi'an 710049, China*
[2]*Shaanxi Provincial Key Laboratory of Big Data Knowledge Engineering, Xi'an Jiaotong University, Xi'an 710049, China*
[3]*School of Distance Education, Xi'an Jiaotong University, Xi'an 710049, China*

Received 16 October 2024/Revised 22 February 2025/Accepted 12 August 2025/Published online 3 November 2025

Abstract Recent advancements in large language models (LLMs) have revolutionized artificial intelligence, yet their computational demands pose significant challenges for efficient deployment. A major issue is handling diverse query responses efficiently, which motivates the need to predict response lengths and optimize the batch processes. In this paper, we thoroughly analyze the challenges involved in the LLM response length prediction task and propose a new framework that treats it as an uncertainty-aware regression problem. We benchmark four uncertainty quantification methods, including both Frequentist and Bayesian approaches, and find that evidential deep learning (EDL) is the most effective and efficient for this task. Furthermore, our case study demonstrates that our approach averagely reduces inference time by 38.14% and 20.50% compared with random batching and the state-of-the-art method, respectively, showcasing the potential of uncertainty-aware response length predictions in optimizing LLM inference.

Keywords large language model, uncertainty quantification, response length prediction

Citation Shi B, Dong B, Zheng Q H. Uncertainty-aware large language model response length perception. Sci China Inf Sci, 2025, 68(11): 210101, https://doi.org/10.1007/s11432-024-4550-8

1 Introduction

Recent advancements in large language models (LLMs) have revolutionized the field of artificial intelligence (AI), demonstrating exceptional capabilities across a broad spectrum of tasks [1, 2], including language translation [3], code generation [4], and even agent-based decision-making [5]. Due to the impressive performance, LLMs are widely applied in an expanding range of domains such as education [6], healthcare [7], and law [8]. However, because of the intensive computational demands of reasoning in LLMs, significant resources are needed to handle diverse query responses, which poses certain challenges for its large-scale deployment [9]. Therefore, how to improve the reasoning efficiency of LLMs and provide users with a good interactive experience has become a key direction for LLM-based AI applications.

In practice, there have been many attempts to improve the efficiency of inference services by designing different task scheduling and batch processing strategies, such as First Come First Server (FCFS) [10], Shortest-Job First (SJF) [11], and multi-level feedback queue-based [12] scheduling. However, since different requests often correspond to different response lengths, sending a batch of inference requests to the LLMs containing sequences with different response lengths will lead to inefficiencies. As requests that were completed earlier are forced to wait for longer requests to complete, this can lead to wasted computation and even cause head-of-line blocking issues [13].

To alleviate this problem, a straightforward solution is to predict the response lengths of given requests. This way, we can group requests with similar lengths in the same batch, thereby optimizing processing efficiency. However, predicting the response length of an LLM is a non-trivial problem. First, there is uncertainty arising from ambiguous training data and input data. For example, the answer to the question "Describe the city you live in." will have high aleatoric uncertainty [14] in training data because it is ambiguous who and when will answer the question. Moreover, LLMs often employ heuristics or sampling methods during response generation (e.g., temperature settings [15], top-k sampling [15]), introducing

* Corresponding author (email: qhzheng@xjtu.edu.cn)

variability in the output. These stochastic elements can lead to responses of differing lengths even for the same input. Thus, there is an urgent need for a new framework that enables response predictions while quantifying the uncertainty of LLMs.

In this paper, we discuss the limitations of several common methods for the perception of LLM response length and propose a new paradigm for this task by incorporating uncertainty quantification. Specifically, instead of relying on traditional deterministic point estimation, we first treat the response length perception problem as an uncertainty-aware regression task that provides confidence intervals for the result. We compare two types of uncertainty quantification methods: Frequentist methods, including quantile regression and mean interval score regression; and Bayesian methods, including Monte Carlo dropout, and evidential deep learning (EDL). We analyze the properties of both Frequentist and Bayesian uncertainty quantification methods as well as their practical performance. We further provide a recipe for practitioners when tackling uncertainty quantification problems in the LLM response length perception task. Moreover, we conducted a case study to evaluate whether the predicted response length could guide the batching strategy and thereby improve the LLM inference efficiency. The results demonstrate the effectiveness of our proposed uncertainty-aware approaches. Specifically, our method averagely reduces the inference time by 38.14% compared with random batching and 20.50% compared with the most up-to-date approaches without uncertainty quantification [16].

Our main contributions are summarized as follows.

• We discuss why uncertainty quantification is important in LLM response length perception tasks and introduce a new paradigm that incorporates uncertainty quantification ability. This new paradigm enables the model to acquire confidence intervals for the predictions. To our knowledge, we are the first to introduce uncertainty quantification in the context of the LLM response length perception task.

• We benchmark four uncertainty quantification methods. Our study reveals that EDL, quantile regression, and mean interval score regression obtain confidence levels that better cover data variations. This observation provides a guide for practitioners in quantifying uncertainty in the LLM response length perception task.

• We conduct a case study, and the result demonstrates that the predicted response length, especially when incorporating uncertainty quantification, provides better guidance for downstream LLM inference serving tasks.

2 Motivation and challenges

2.1 Serving LLM applications

Growing demands for LLM applications have made the LLM inference serving engine a critical component in modern datacenters. End-users or other microservices of LLM applications submit requests to the LLM inference serving engine, which queues the incoming requests, dispatches jobs to available computing devices such as GPUs or TPUs, and returns the results to the end-users. Like other scheduling engines in datacenters, a well-managed inference service should provide low latency and high throughput within a reasonable amount of cost.

To achieve this goal, existing serving engines such as TensorFlow serving [17] typically batch requests to increase hardware utilization and system throughput by better exploiting parallel computing units in hardware accelerators. However, in practice, such batching and scheduling strategies face a serious challenge due to the significant variation in response lengths for different queries. As demonstrated in Figure 1, the LLM response lengths of 10k queries in the Alpaca dataset [18] distribute across a wide range. This problem originates from the autoregressive nature of the LLM: the models are trained to generate subsequent tokens iteratively, continuing the process until the end of the token sequence. In other words, unlike traditional deep neural network (DNN) models, such as ResNet [19] or MLP [20], which have fixed inference costs [21], the total number of iterations and execution time for an LLM are unknown in advance for a given input query. As a result, when sending a batch of inference requests to an LLM, the inclusion of sequences with differing response lengths will lead to inefficiencies. Requests that have been completed earlier are forced to wait for longer ones to complete, resulting in computational waste and even head-of-line blocking problems [13]. We show an experiment about this problem in Figure 2, where the time spent waiting for a shorter answer length is wasted computation. Similarly, we selected 10k queries and packaged them into batches of different sizes, including 4, 8, 16, 32, and 64, for

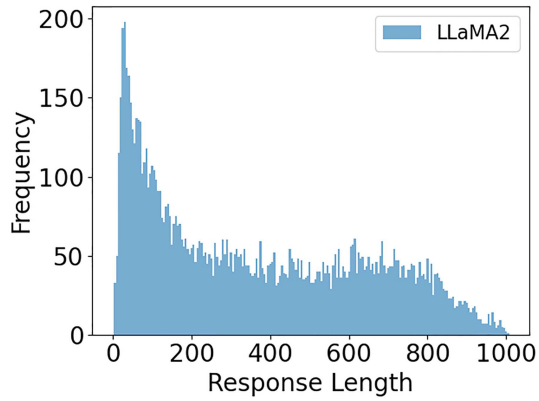

Figure 1 (Color online) Distribution of response length.

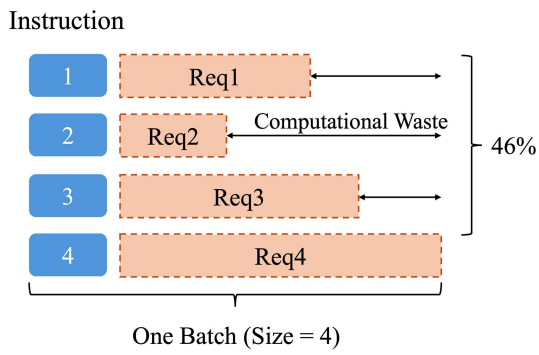

Figure 2 (Color online) Computational waste when long and short responses are in the same batch.

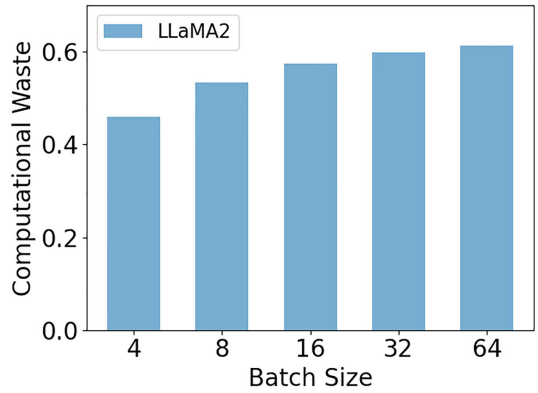

Figure 3 (Color online) Computational waste vs. batch size.

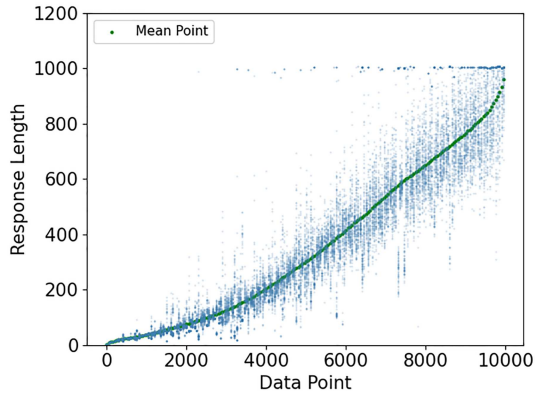

Figure 4 (Color online) Distribution of mean length among 120 times generations on 10k instructions.

the experiment. The results are shown in Figure 3. It can be seen that the waste of computing resources increases with increasing batch size, reaching nearly 60% at 64. This issue constitutes the motivation for this study. If we can predict the response length of each request, we can group requests with similar lengths in the same batch, thereby effectively minimizing computational waste.

- **Takeaway 1:** Perception of LLM response lengths could help improve LLM inference efficiency.

2.2 Response length perception

2.2.1 *The problem of point estimation*

Typically, the LLM response length perception task will be formulated as a point estimation task, where the objective is to predict a specific response length for a given query. However, as discussed in Section 1, the response length to identical queries can differ across various samples. Figure 4 illustrates the variability in response length, and we sample 10k instructions from the Alpaca dataset [18], prompting the LLaMA2-7B to answer each query 120 times. The figure plots the range of possible lengths for each query (from the shortest to the longest response). In the figure, we can see that there is no definitive 'ground truth' but rather a range of possible truths for LLM response length perception tasks. Therefore, the traditional deterministic point estimation framework may feel confused to make a decision, since any given answer could be both correct and incorrect at the same time. It is urgent to measure the uncertainty in this setting and thereby provide confidence intervals for the result.

- **Takeaway 2:** Point estimation is not suitable for the LLM response length perception tasks; instead, uncertainty quantification is needed.

	Non-intrusive Manner	Intrusive Manner
Zero-shot Prompt	***Instruction*** Don't output the response for the following input. Instead, you need to predict the length range of words in your response. Output two numbers only. ***Input*** Provide a list for the 7 wonders of the ancient world. ***Output*** Length range of words: 7-10.	***Instruction*** Before responding to the following input, you need to predict the length range of words for your response in the first line. **Then change to a new line to respond to the input.** ***Input*** Provide a list for the 7 wonders of the ancient world. ***Output*** Sure! Here are the 7 wonders of the ancient world, along with their estimated lengths:1. Great Pyramid of Giza - 1,000 feet (305 meters)2. Hanging Gardens of Babylon - 300 feet (91 meters)3. Statue of Zeus at Olympia… ✗
Few-shot Prompt	***Instruction*** Don't output the response for the following input. Instead, you need to predict the length range of words in your response. Output two numbers only. ***Demonstrations*** There are *K* demonstrations: *1. Input: Generate a potential business name related to pet food industry. Output: My response will likely be around 5-10 words long.* *2. Input: Suggest a word to replace "stupid". Output: My response will likely be around 5-15 words long.* ... ... ***Input*** Provide a list for the 7 wonders of the ancient world. ***Output*** My response will likely be around 10-15 words long.	***Instruction*** Before responding to the following input, you need to predict the length range of words for your response in the first line. **Then change to a new line to respond to the input.** ***Demonstrations*** There are *K* demonstrations: *1. Input:Generate a potential business name related to pet food industry. Output:Word range: 2-5 words \n **PetPal Feeds**.* *2. Input:Suggest a word to replace "stupid". Output:Word range: 1-2 words \n **Unwise**.* ... ... ***Input*** Provide a list for the 7 wonders of the ancient world. ***Output*** Word range: 3-5 words \n **Great Pyramid of GizaHanging Gardens of Babylon Statue of Zeus at Olympia Temple of Artemis at Ephesus Mausoleum at Halicarnassus Walling of Thebes Colossus of Rhodes.**

Figure 5 (Color online) Examples of four different prompts to predict the length of the responses on LLaMA2-7B. As demonstrated in the upper right corner, LLaMA2-7B may fail to follow the instructions.

2.2.2 *Response length perception by LLM itself*

Humans have the ability to estimate the length of an answer based on their understanding of the question. For example, a question like "What is the capital of China?" typically receives a shorter response than a question like "Please introduce the landmarks of Beijing". Since LLMs have undergone reinforcement learning from human feedback (RLHF), they have, to some extent, been aligned with human understanding. Thus, they naturally exhibit the emergent ability [22] of response length perception. In this subsection, we evaluate how well this ability is developed. Specifically, we directly asked the LLMs to predict the length of the responses they were about to generate. Note that instead of instructing the model to output a point estimation, we have it consider the uncertainty and output a range. Furthermore, we did not consider advanced prompt design techniques such as chain of thought [23], scratchpad [24], focusing solely on zero-shot and few-shot prompts for testing. The specific settings include:

Non-intrusive manner. This process involves modifying the query to prompt the model to provide only an estimated range of response lengths. We then compare this estimated length with the actual response generated using the original prompt. In other words, this approach decouples the prediction and generation processes.

Intrusive manner. This process involves modifying the query to ask the model to answer the estimated range of response lengths first and then provide the full response.

Figure 5 illustrates examples of four different prompts designed to make LLMs predict the length of the responses they were about to generate. We applied these modified prompts to both GPT-4 and LLaMA2-7B and observed how they responded to the instructions. Specifically, we modified 200 queries from the Alpaca dataset and employed the metric *Accuracy* to evaluate whether the actual response lengths[1)] for these queries accurately fell within the predicted ranges of the different models. In addition, we recorded the average interval widths of the predicted ranges for each model. The results are shown in Table 1. Bold and underline in the table indicate the best results within the same group. As shown, regardless of the type of large model, whether few-shot or zero-shot, the prediction accuracy in a non-intrusive manner is low. In terms of intrusive manner, we observed that LLaMA2-7B yielded low performance, with instances where the model failed to follow the prompt instructions. This behavior can be attributed to the limited capacity of smaller LLMs to handle multiple tasks simultaneously. Additionally, when using few-shot prompts, the numbers in the examples can be somewhat misleading for LLaMA2-7B's predictions, as the model has limited capacity to understand the examples and tends to directly repeat the numbers mentioned in the prompt. In contrast, larger LLMs like GPT-4 exhibited good performance, particularly

1) In this experiment, the actual response lengths were only sampled once.

Table 1 Evaluation of response length perception across four different prompts.

Metrics	Zero-shot prompt		Few-shot prompt	
	Interval width	Accuracy (%)	Interval width	Accuracy (%)
		Non-intrusive manner		
LlaMA2-7B	26.17	26.5	6.21	26.0
GPT-4	39.34	26.5	16.30	35.5
		Intrusive manner		
LlaMA2-7B	23.50	21.0	1.33	20.5
GPT-4	14.28	**58.5**	18.00	**67.5**

Table 2 Evaluation of response length perception under instruction tuning.

	Interval width	Accuracy (%)
LLaMA2-7B without tuning	26.17	26.5
LLaMA2-7B with tuning	619.6	90.5

when using few-shot prompting, achieving an accuracy of 67.5%. However, the intrusive approach has a significant drawback, as the model is aware of the anticipated length during the creation of the response. This awareness can lead the model to interpret the estimated length as a boundary condition, potentially shaping its output to conform to the predicted length, which can be referred to as a length-constrained generation process. We analyze the cases and observe a stronger tendency for GPT-4 to tailor its answers to fit the estimated length.

• **Takeaway 3:** Directly prompting LLMs to predict the response length without adjusting or fine-tuning them is not feasible.

2.2.3 *Response length perception through instruction-tuning*

Based on Takeaway 3, we further investigate the ability of LLMs to predict the length of their responses through instruction-tuning.

Instruction-tuning manner. This process involves supervised instruction tuning. We modify the input query and prompt the model to predict the range of the response length instead of generating the entire response. Specifically, we add "Do not output the response for the following instruction. Instead, you need to predict word count range of your response. Output two numbers only, for example, 300–500" in front of the original query. Then, a subset of 10000 queries from the Alpaca dataset [18] was selected as input, and the labels were set according to the minimum and maximum lengths observed over 120 generations. To minimize the computational resource required during instruction-tuning, we employ the efficient training method LoRA [25].

Table 2 presents the experimental results, which demonstrate a significant improvement after fine-tuning. The prediction accuracy increased from 26.5% to 90.5%. Although such results have proven the effectiveness of instruction tuning, we believe that this approach is not optimal for the following reasons. (1) Even with the use of LoRA, instruction tuning still requires a significant amount of training resources. (2) The fine-tuned LLMs make predictions in a generative manner, which may not be ideal for perception tasks such as length prediction in terms of both accuracy and efficiency [26]. The results analysis also corroborates this point: we observe that the fine-tuned model tends to predict the minimum length close to 0, which is not a precise answer.

• **Takeaway 4:** Supervised instruction-tuning can improve the accuracy of length prediction, but it comes with significant training and prediction overhead. Given that the output dimension for the response length prediction task is very low, a lighter model should be sufficient.

3 Method

In this section, we propose the Uncertainty-aware LLM response length perception framework. As shown in Figure 6, we froze the LLM and trained a separate model to perform the length prediction task in an uncertainty-aware manner. During the training (or prediction) phase, we fed the model with the hidden states, which were selected at the point when the LLM encoded the last token of the input query. This

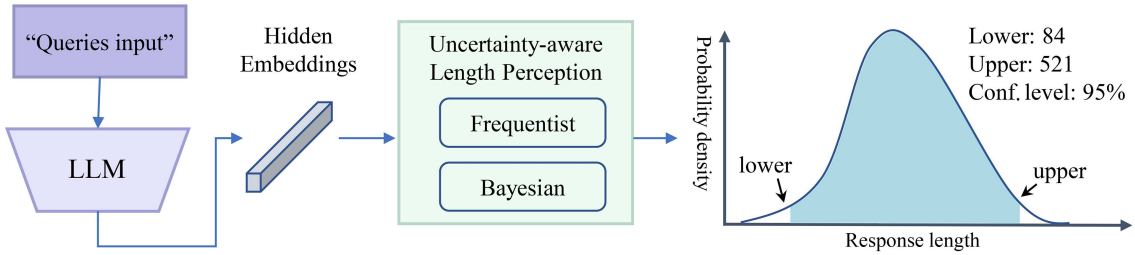

Figure 6 (Color online) The overview of uncertainty-aware large language model response length perception.

approach resulted in minimal loss in our experiments, outperforming methods such as average pooling or concatenating the hidden states of multiple or all tokens. The specific task definition is as follows.

3.1 Problem definition

The prediction of response length naturally fits into the regression framework, where the goal is to predict a continuous variable. Given a data set $D = \{x_i, y_i\}_{i=0}^{N}$, where x_i is the hidden state embedding of the last token in the last layer of the ith prompts [27] and $y_i \in \mathbb{R}$ is the corresponding true length label. The goal is to find a function f_θ with deep neural networks:

$$f_\theta(x_i) = \hat{y}_i, \tag{1}$$

where θ is the function parameter and $\hat{y}_i$ denotes the predicted response length.

The function parameter θ is estimated by minimizing the loss between the predicted and real response length:

$$S(\theta) = \frac{1}{N}\sum_{i=0}^{N} \mathcal{L}\ (y_i, \hat{y}_i), \quad \min_\theta S(\theta). \tag{2}$$

$\mathcal{L}(\cdot)$ is the loss function, such as the mean squared error (MSE), the mean absolute error (MAE). In this study, we choose the MSE function:

$$\mathcal{L}\ (y_i, \hat{y}_i) = \|y_i - f_\theta(x_i)\|^2. \tag{3}$$

However, the traditional regression problem mentioned above only considers point estimation and cannot reflect the uncertainty in the data and model. In reality, due to uncertainty, there are inevitably multiple different response lengths for the same prompt input. Let $Y_i = \{y_{i1}, y_{i2}, y_{i3}, \ldots, y_{in}\}$ denote all possible response lengths. In this paper, we redefine the task for LLM Response Length Perception that incorporates uncertainty quantification ability. The objectives are as follows:

$$\begin{aligned} &\min\ \ width(\mathcal{C}_i) \\ &\text{s.t. } \mathbb{P}\{Y_i \in \mathcal{C}_i\} \geqslant 1-\alpha, \end{aligned} \tag{4}$$

where $\mathcal{C}_i$ denotes the confidence interval, α is the significance level[2)]. The $width(\mathcal{C}_i)$ denotes width of confidence interval $\mathcal{C}_i$.

Eq. (4) aims to achieve coverage while minimizing the confidence interval width. Coverage: The confidence interval $\mathcal{C}_i$ covers the possible response lengths with a confidence level of $1-\alpha$. Width: The width refers to the range of confidence interval $\mathcal{C}_i$. The former measures how often the true target lies within the confidence interval, while the latter measures the precision of the provided confidence interval. Ideally, we aim to achieve high coverage while maintaining minimal width.

Based on the task formulation, we further apply statistical decision theory to integrate Frequentist and Bayesian methods into a unified framework for better quantifying the uncertainty. We assume that dataset $\mathcal{X}$ and its corresponding labels $\mathcal{Y}$ follow the probability distribution $p(\mathcal{Y} \mid \mathcal{X}; \theta)$ with parameter θ, and θ also follows distribution $p(\theta)$. The objective of the training process is to minimize the expected loss function over both the data distribution and the model parameter distribution $p(\theta)$. The difference is, the Frequentist methods assume that the parameter θ is fixed but unknown, interpreting probability

2) In this paper, we employ α to denote the significance level and $\rho = 1-\alpha$ to denote the confidence level.

as the frequency of an event occurring over an infinite number of repeated experiments. These methods use hypothesis testing and confidence intervals to determine $\mathcal{C}_i$. In contrast, Bayesian methods treat the parameter θ as a random variable and follow a prior distribution. By applying Bayes' theorem, the prior distribution is combined with the likelihood function derived from the observed data to obtain the posterior distribution, from which confidence intervals $\mathcal{C}_i$ are derived. Next, we detail how to estimate confidence intervals for LLM response length perception using these uncertainty quantification methods.

3.2 Frequentist UQ methods

Mean interval score regression (MISR). MISR [28] can be employed to estimate confidence intervals in a regression problem through a single forward pass. Specifically, given confidence level $1-\alpha$, we use the multi-head MLP to output upper confidence bound $u(x)$, lower confidence bound $l(x)$, and the prediction $f(x)$. It tends to output narrower confidence intervals and encourages intervals that include the true target (coverage). We optimize MISR by minimizing the following loss:

$$\begin{aligned} &L_{MISR}(y, u(x), l(x), f(x); \theta, \alpha) \\ &\quad = \min_{\theta} \left\{ \mathbb{E}_{(x,y)\sim\mathbb{D}} \left[(u(x) - l(x)) + \frac{2}{\alpha}(y - u(x))\mathbb{I}\{y > u(x)\} + \frac{2}{\alpha}(l(x) - y)\mathbb{I}\{y < l(x)\} + |y - f(x)| \right] \right\}, \end{aligned} \tag{5}$$

where $\mathbb{I}\{\cdot\}$ represents the indicator function, which returns 1 when the condition is true. $(x, y) \sim \mathbb{D}$ indicates that the random variable (x, y) follows a joint distribution $\mathbb{D}$.

Quantile regression (QR). QR [29] can obtain the corresponding quantile of input value x. Given a confidence interval $(1 - \alpha)$, by regressing multiple quantile points such as $(1 - \frac{\alpha}{2})$ and $\frac{\alpha}{2}$ quantiles simultaneously during one forward propagation process, the corresponding confidence interval upper and lower bounds with $(1 - \alpha)$ confidence interval can be directly obtained,

$$\begin{aligned} &L_{QR}(y, f(x); \theta, \tau) \\ &\quad = \min_{\theta} \{ \mathbb{E}_{(x,y)\sim\mathbb{D}} [(1 - \tau)|y - f(x)|\mathbb{I}\{f(x) < y\} + \tau|f(x) - y|\mathbb{I}\{f(x) > y\}] \}, \end{aligned} \tag{6}$$

where $\mathbb{I}\{\cdot\}$ represents the indicator function, which returns 1 when the condition is true. $(x, y) \sim \mathbb{D}$ indicates that the random variable (x, y) follows a joint distribution $\mathbb{D}$. The τ is the quantile point, which varies according to the given confidence level, taking on the values $[\frac{\alpha}{2}, 0.5, 1 - \frac{\alpha}{2}]$, respectively.

3.3 Bayesian UQ methods

Monte-Carlo Dropout (MC Dropout). MC Dropout [30] is widely used to quantify the uncertainty of neural network models. This method achieves uncertainty quantification by activating the Dropout layer in the inference phase and performing multiple forward propagations to achieve approximate Bayesian inference.

In general, MC Dropout can be applied without restrictions, requiring only the dropout layer in the neural network model. During the training stage, MC Dropout is the same as a regular neural network. During the test stage, the dropout layer is required to be activated. When MC Dropout is running, each time a forward propagation is performed, a random Monte Carlo sampling is completed. Based on the results of multiple forward propagations, we can calculate the prediction mean and variances, and then estimate the confidence interval. In addition, MC Dropout can be implemented in parallel, and when there are sufficient computing resources, the time cost is equivalent to one forward propagation.

Evidential deep learning (EDL). EDL [31] represents the learning as an evidence acquisition process, which does not place the prior on the network weights as in the case of a Bayesian neural network, but directly places the prior on the likelihood function. By training the neural network to output hyperparameters for high-order evidence distributions, we can obtain a basic representation of uncertainty in discrete classification tasks without sampling. In this paper, we mainly introduce the variants of evidential deep regression (EDR) [32] that are suitable for regression tasks. We assume that label y_i follows a Gaussian distribution with unknown mean and variance (μ, σ^2). In order to model this, we also need to perform prior estimation on them separately. When assuming that the predicted label

follows a Gaussian distribution, this leads to μ and σ^2 satisfying the Gaussian prior and Inverse-Gamma prior, respectively.

$$(y_1, y_2, \ldots, y_N) \sim N(\mu, \sigma^2), \tag{7}$$

$$\mu \sim N(\gamma, \sigma^2 \nu^{-1}), \quad \sigma^2 \sim \Gamma^{-1}(\alpha, \beta), \tag{8}$$

where $\Gamma(\cdot)$ is Gamma function, $m = (\gamma, \nu, \alpha, \beta)$, $\gamma \in \mathbb{R}$ and $\nu > 0$, $\alpha > 1$, $\beta > 0$.

We aim to calculate the posterior distribution $q(\mu, \sigma^2) = p(\mu, \sigma^2 \mid y_1, y_2, \ldots, y_N)$, which can be factorized [33] into $q(\mu, \sigma^2) = q(\mu)q(\sigma^2)$, so we can use the normal inverse-gamma (NIG) distribution:

$$p(\mu, \sigma^2 \mid \underbrace{\gamma, \nu, \alpha, \beta}_{m}) = \frac{\beta^\alpha}{\Gamma(\alpha)\sqrt{2\pi\sigma^2}} \left(\frac{1}{\alpha^2}\right)^{\alpha+1} \exp\left\{ - \frac{2\beta + \nu(\gamma - \mu)^2}{2\sigma^2} \right\}. \tag{9}$$

In order to infer the hyperparameters m of the high-order evidence distribution under a given input, EDR constructs the learning process into two parts, maximizing the model evidence support the observations, where the model evidence is defined as the likelihood of y_i given the evidence distribution hyperparameters m:

$$\begin{aligned} p(y_i \mid m) &= \int_{\sigma^2=0}^{\infty} \int_{\mu=-\infty}^{\infty} p(y_i \mid \mu, \sigma^2) p(\mu, \sigma^2 \mid m) d\mu d\sigma^2 \\ &= St\left(y_i; \gamma, \frac{\beta(1+\nu)}{\nu\alpha}, 2\alpha\right), \end{aligned} \tag{10}$$

where $St(y; \mu, \sigma^2, n)$ is the the Student-t distribution at y with location μ, scale σ^2 and degree of freedom n.

The model outputs the maximum model evidence by minimizing the negative log marginal likelihood loss (NLL) for each training pair (x_i, y_i), $L_{\mathrm{NLL}}(y_i, m) = -\log(p(y_i \mid m))$, which can be summarized as the following formula:

$$\begin{aligned} &\min_\theta \{L_{\mathrm{NLL}}(y, m)\} \\ &\quad = \min_\theta \left\{ \mathbb{E}_{(x,y)\sim\mathbb{D}} \left[\frac{1}{2} \log\left(\frac{\pi}{\nu}\right) - \alpha \log(\Omega) + \left(\alpha + \frac{1}{2}\right) \log((y-\gamma)^2\nu + \Omega) + \log\left(\frac{\Gamma(\alpha)}{\Gamma(\alpha + \frac{1}{2})}\right) \right] \right\}, \end{aligned} \tag{11}$$

where $\Omega = 2\beta(1 + \nu)$.

When the model makes incorrect predictions, an incorrect evidence penalty is used to regulate the training process to minimize the evidence of incorrect predictions:

$$\min_\theta \{L_R(y, m)\} = \min_\theta \{\mathbb{E}_{(x,y)\sim\mathbb{D}}[|y - \mathbb{E}(\mu)| \cdot \Phi]\} = \min_\theta \{\mathbb{E}_{(x,y)\sim\mathbb{D}}[|y - \gamma| \cdot (2\nu + \alpha)]\}. \tag{12}$$

According to the high-order evidence parameters $m = \{\gamma, \nu, \alpha, \beta\}$ output by the EDR model, the model prediction $\mathbb{E}[\mu]$, aleatoric uncertainty $\mathbb{E}[\sigma^2]$, and epistemic uncertainty $\mathrm{Var}[\mu]$ can be calculated by the following formula:

$$\mathbb{E}[\mu] = \gamma, \quad \mathbb{E}[\sigma^2] = \frac{\beta}{\alpha - 1}, \quad \mathrm{Var}[\mu] = \frac{\beta}{\nu(\alpha - 1)}. \tag{13}$$

3.4 Implementation details

We implement these models using PyTorch. All models consist of two fully connected layers and one output layer, with an activation function applied after each layer. The number of hidden units in each layer is set to 60. Additionally, we use a dropout layer to prevent overfitting, with a dropout rate of 0.1. The parameter counts for each layer are as follows. Layer 1: $\mathcal{P}_1 = d_{\mathrm{input}} \times 60 + 60$. Layer 2: $\mathcal{P}_2 = 60 \times 60 + 60$. Output: $\mathcal{P}_3 = 60 \times d_{\mathrm{output}} + d_{\mathrm{output}}$. We use the Adam optimizer, set the learning rate to 1×10^{-7}, and train for 50 epochs with a batch size of 32.

4 Experiments

4.1 Experimental setting

Two datasets, the Alpaca dataset [18] and the Instruction-in-Wild dataset [34], are included in our experiment. The first one contains 52k query instructions generated based on the self-instruction framework, from which we randomly select 10k instructions for our experiment. The second one contains 429 real-world collected query instructions, and we use them all.

In all experiments, we consistently employed two base models, LLaMA2-7B [35] and Qwen2.5-14B-Instruct [36], to generate hidden state embeddings. For each query, both models were run 120 times to obtain a tuple of response length as the prediction label. The set temperature is 0.6, $top\_p = 0.9$, and $max\_seq\_len = 1024$. During the training process, separate training processes were conducted for both LLaMA2-7B and Qwen2.5-14B-Instruct. The training dataset, validation dataset, and test dataset are divided into 8:1:1. We choose the epoch with the smallest MAE loss as the optimal model and the 95% confidence interval as the default value for calculation. For ECE, the confidence interval $\rho_i = \{0.1, 0.2, \ldots, 0.9\}$. For QR, we set the quantile to $\{\frac{\rho}{2}, 0.5, (1-\frac{\rho}{2})\}$ and can obtain the mean value, upper, and lower bounds in one propagation. For MC Dropout, we sample 100 times, with a dropout rate of 10%. The setting of EDL is the same as that in [37]. For instruction tuning, we also tuned the LLM to predict the response length in a point estimation manner.

4.2 Evaluation metrics

For point estimation in regression tasks, the mean absolute error (MAE) is a commonly used metric. However, when dealing with predictions with uncertainty, MAE cannot fully represent the quality of the prediction interval. Therefore, we discuss several alternative metrics.

Mean interval score (MIS). MIS is used to measure the prediction interval, also known as the Winkler loss [38]. It encourages minimizing the prediction interval and rewards intervals that cover the true label. For a given input value x and the corresponding confidence interval $(1-\alpha)$, if the upper and lower bounds of the interval are u and l, then the calculation of MIS is as follows:

$$MIS_N(u, l; \alpha) = \frac{1}{N}\sum_{i=0}^{N}\left\{(u_i - l_i) + \frac{2}{\alpha}(y_i - u_i)\mathbb{I}\{y_i > u_i\} + \frac{2}{\alpha}(l_i - y_i)\mathbb{I}\{y_i < l_i\}\right\}. \tag{14}$$

Interval score (IS). IS [39] measures the average size of the prediction confidence intervals. Given confidence interval ρ, IS is calculated based on the corresponding upper and lower bounds, u and l. The detailed definition is as follows:

$$IS_N(u, l; \rho) = \frac{1}{N}\sum_{i=0}^{N}(u_i - l_i). \tag{15}$$

Expected calibration error (ECE). ECE [37,40,41] measures how well the predicted probabilities (or confidence intervals) of a model align with the actual observed frequencies of the true target:

$$ECE = \frac{1}{N}\sum_{i=0}^{N}|acc(\rho_i) - \rho_i|, \tag{16}$$

where ρ_i denotes the confidence interval, $acc(\rho_i)$ represents the proportion of true labels that fall within the corresponding confidence interval $[\frac{\rho_i}{2}, 1-\frac{\rho_i}{2}]$. In this paper, we use $\rho_{[0:N]} = [0.01, 0.02, \ldots, 1.00]$ enumerably, where $N = 100$.

Coverage rate. Coverage rate measures the average proportion of actual response lengths (true labels) that accurately fall within the predicted ranges. Since we sampled 120 response lengths for each query, we represent this value as $x/120$, where x denotes the average number of predicted response lengths that fall within the corresponding confidence interval.

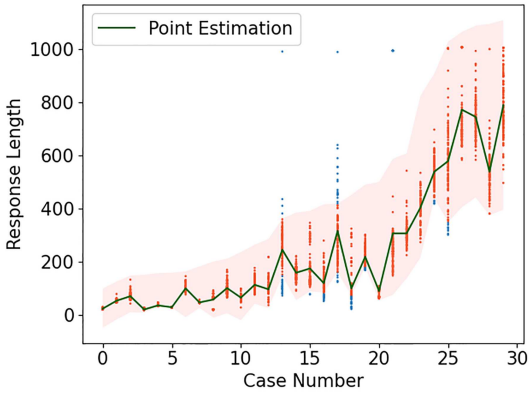

Figure 7 (Color online) Case visualization on Alpaca.

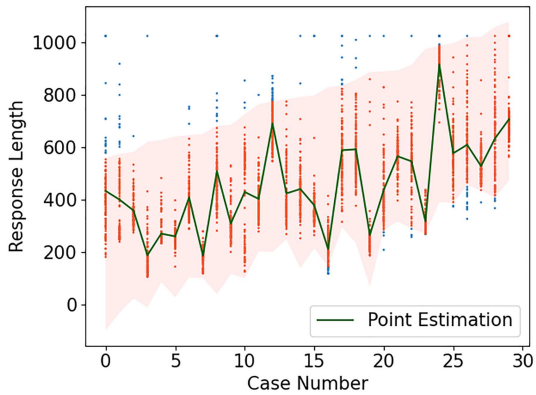

Figure 8 (Color online) Case visualization on Instruction-in-Wild.

4.3 Performances and analysis

Table 3 shows the main results obtained from testing on two different datasets using both LLaMA2-7B and Qwen2.5-14B-Instruct models. First, we can see that the Instruction Tuning method generally shows lower performance than uncertainty quantification methods across all metrics, particularly in terms of inference and training times. As for uncertainty quantification methods, the QR and EDL methods exhibit fine predictive performance in the MAE metric, even better than simple point estimation prediction methods. In addition, MISR, QR, and EDL methods have similar performance scores in the MIS, IS, and ECE metrics. For the MIS score, MISR directly minimizes the MIS score function but only achieves the best performance on the Alpaca dataset when using LLaMA2-7B, while EDL performs best on the remaining settings. Furthermore, the EDL method also achieves optimal performance on both two datasets in the ECE metric for both models, indicating that the EDL has the optimal uncertainty quantification ability and its predictions align more closely with the true distribution of the data. In comparison, MC Dropout may struggle to fully capture the diversity of the data and tends to produce narrower confidence intervals than desired, which is expected since MC Dropout is designed to measure epistemic uncertainty, while the context here is dominated by aleatoric uncertainty. In terms of efficiency, MISR, QR, and EDL have negligible additional time costs compared with simple point estimation. Although MC Dropout has a similar time cost to point estimation during the training process, it significantly increases the overhead during the testing process due to the need for multiple samples. In conclusion, we consider EDL to be the best choice in terms of effectiveness and efficiency.

4.4 Visualization

To illustrate the necessity of introducing uncertainty quantification into LLM response length perception, we conduct a case study visualization in two datasets, respectively, see in Figures 7 and 8. We randomly select 30 results from each dataset on the LLaMA2-7B model, where the points are the true response length distribution for different queries, the green lines represent point estimation results, and the shades represent the estimated confidence interval 95%. We also use the same color to mark those true response length points that fall within the confidence intervals. As can be seen from the two figures, the simple point estimation method can only obtain the average value of the true distribution and is easily affected by extreme values (too large or too small). In contrast, the uncertainty quantification method we introduce can better capture the uncertainty in the dataset and the way LLM itself generates by giving a confidence interval, which is more suitable for the response length perception task of LLM.

4.5 Interesting findings

During our investigation into the length of LLM responses, we made some interesting observations regarding what types of prompts are more likely to generate longer responses.

(1) Long responses with low variability.

(i) **List or enumeration tasks** (e.g., "Generate a 10-item annotated bibliography given the information below.") consistently produced the longest responses. This is mainly because list-based tasks require

Table 3 We compare point estimation, instruction tuning, and four uncertainty quantification methods across four metrics: MAE, MIS, IS, and ECE. Additionally, we also compare the training and testing time required for each method.

		Alpaca dataset						
		MAE	MIS	IS	ECE	Coverage rate	Train time (s)	Test time (s)
LLaMA2-7B	Point estimation	121.2751	–	–	–	–	45.4446	3.6216
	Instruction tuning	137.3568	1022.8373	619.6240	–	109.07/120	3305.4856	112.2594
	Frequentist methods							
	MISR	130.7789	590.4245	435.8109	0.0706	112.81/120	104.8951	3.8201
	QR	121.3691	611.5578	400.6010	0.0681	112.12/120	119.8766	5.6572
	Bayesian methods							
	MC Dropout	125.3334	1518.5341	151.1300	0.1515	53.40/120	40.44456	55.0553
	EDL	119.2315	602.7090	420.0773	0.0610	114.57/120	155.8865	3.3970
Qwen2.5-14B	Point estimation	113.8544	–	–	–	–	49.1924	3.2231
	Instruction tuning	130.1769	984.6346	591.4495	–	111.83/120	11881.1375	314.8146
	Frequentist methods							
	MISR	126.4136	568.7523	412.3513	0.0723	114.90/120	129.9201	4.6501
	QR	115.2269	588.3482	381.9124	0.0698	114.05/120	136.2043	4.2023
	Bayesian methods							
	MC Dropout	122.1772	1459.2896	138.7250	0.1571	56.33/120	79.1925	52.6465
	EDL	113.5837	581.0357	405.1995	0.0635	115.42/120	160.8388	2.9110
Metrics		Instruction-in-Wild dataset						
		MAE	MIS	IS	ECE	Coverage rate	Train time (s)	Test time (s)
LLaMA2-7B	Point estimation	172.0837	–	–	–	–	2.7644	0.2329
	Instruction tuning	189.2764	1176.5384	683.4197	–	105.84/120	96.7815	14.3547
	Frequentist methods							
	MISR	195.9512	954.3300	591.3186	0.0601	105.77/120	6.2523	1.3882
	QR	171.4666	819.5064	652.7430	0.0566	107.38/120	3.7398	0.3775
	Bayesian methods							
	MC Dropout	186.4899	1878.9079	317.8341	0.0812	76.97/120	2.3403	4.7700
	EDL	163.1294	709.2047	560.6721	0.0420	110.04/120	6.7090	0.2711
Qwen2.5-14B	Point estimation	164.5589	–	–	–	–	3.1265	0.2952
	Instruction tuning	171.4472	1083.6251	634.3585	–	107.92/120	680.6921	108.4473
	Frequentist methods							
	MISR	178.6653	691.3849	547.3265	0.0612	110.35/120	6.4521	1.0128
	QR	165.3274	723.2189	515.3265	0.0573	111.84/120	2.9065	0.4297
	Bayesian methods							
	MC Dropout	175.9921	1572.4385	275.3265	0.898	58.71/120	2.9110	4.3453
	EDL	159.4376	635.2189	532.6532	0.0487	112.93/120	6.9973	0.3067

the model to enumerate and explain each item individually, resulting in long but structurally consistent outputs.

(ii) **Multi-part tasks** (e.g., "Look up the definition of the Latin phrase 'lexicalis', and explain it in your own words.") generated predictably long responses. In these tasks, the sequential steps or required components are explicitly defined. The fixed substructure constrained the length variation while making the response longer.

(2) Long responses with high variability.

(i) **Open-ended prompts** (e.g., "Explain a common misconception about your topic.") generated responses that were both long and highly variable. These prompts require the model to synthesize broader knowledge, elaborate abstract ideas, and sometimes include optional examples. For example, responses to such prompts were on average 2.3 times longer than those to factual Q&A (e.g., "What is the capital of France?"), but exhibited 25%–40% variation in length depending on how expansively the model chose to elaborate.

(ii) **Creative generation prompts** (e.g., "Create a story for a children's book about a crocodile.") showed the highest length variability among all categories. The open-ended nature of creative writing, coupled with the need for narrative development—such as plot, character, and setting—resulted in outputs that were often lengthy, but with highly inconsistent lengths depending on the narrative scope the

Table 4 Comparison of throughput across different scheduling strategies.

	LLaMA2-7B		Qwen2.5-14B	
	Throughput (samples/s)	Improvement (%)	Throughput (samples/s)	Improvement (%)
Vanilla	0.413	0	0.174	0
Zheng2024	0.454	9.93	0.209	20.11
EDL+Upper	0.579	40.19	0.231	32.76
EDL+Lower	0.557	34.87	0.215	23.56
EDL+Expected	0.562	36.08	0.222	27.59
EDL+Cluster	0.588	42.37	0.233	33.91

model chose to develop.

These findings highlight that response length is highly sensitive to prompt specificity. At the same time, they underscore the importance of predicting response lengths when serving LLM applications.

5 Case study: improve LLM inference efficiency

After developing a precise response length perception module, this section presents case studies to demonstrate how uncertainty-aware length prediction improves the efficiency of LLM inference. To enable efficient sequence scheduling, we assume that the group size (i.e., the number of queries that need to be processed) is larger than the batch size for a single GPU. This assumption aligns with the common scenario of LLMs. As shown in Figure 2, batching queries with vastly different response lengths leads to redundant computations, reducing inference throughput. By grouping queries with similar response lengths, we can significantly accelerate the inference process. However, based on the analysis in this paper, the response lengths are uncertain and cannot be precisely determined, so it is challenging to group them in advance.

Case setting. In our setting, the predicted response length includes an upper bound u, a lower bound l, and an expected value $\hat{y}_i$. Therefore, we adopt 3 basic grouping strategies: (1) sorting the queries according to their predicted upper bound (EDL+Upper), lower bound (EDL+Lower), and expected value (EDL+Expected), respectively; (2) sequentially dividing them into batches for processing. In addition, we also propose the use of clustering methods to achieve more effective grouping (EDL+Cluster). The clustering parameters are set as $[\frac{u+l}{2}, u-l]$, and the clustering algorithm used is K-means [42]. The baseline of our experiment is (1) Vanilla: the queries are randomly batched for inference; (2) Zheng2024 [16]: using instruction tuning for maximum response length prediction and then batching queries by the predictions. In the evaluation, we randomly selected samples from the Alpaca dataset. We defined the query group size as 256 and the batch size as 16. The inference is performed on both the LlaMA2-7B and the Qwen2.5-14B model using an 80GB A100 GPU.

Overhead of length prediction. Given that the proposed method requires computing the contextual embeddings for all queries before generating their responses, there is an inherent overhead associated with response length prediction. This overhead primarily involves calculating the key-value (KV) cache for the instruction tokens (referred to as pre-filling). However, this overhead is actually minimal, as the process typically involves a single forward pass, allowing the model to process all input sequences in parallel. For instance, in Transformer models, tokens in the input sequence are computed simultaneously via matrix multiplication, thereby fully leveraging the parallel computing capabilities of GPUs/TPUs. Our experiments using LLaMA2-7B on the Alpaca dataset revealed that this process typically requires a computational time comparable to generating 1–2 tokens. Our benchmarks on LLaMA2-7B (A100 GPU) show that prefilling a 512-token query takes 31.31 ms, which is comparable to generating 1.12 tokens (27.93 ms/token). Consequently, when performing batching generation with similar lengths queries, we re-perform the prefilling operation instead of reusing the previously calculated KV cache. Furthermore, the minimal overhead associated with this process can be effectively compensated by the overall acceleration achieved through sequence scheduling.

Results. Table 4 presents the throughput across different scheduling strategies. It is worth noting that all associated overheads, such as response length prediction and grouping, are included in the experiments. Among our 4 different methods, the clustering-based batch grouping approach (EDL+Cluster) demonstrates the best performance, achieving throughput improvements of 42.37% on LLaMA2-7B and 33.91% on Qwen2.5-14B compared with the baseline (Vanilla) method. Furthermore, our method out-

Table 5 Performance of model trained at $temperature = 0.6$, $top\_p = 0.9$ and evaluated on random $temperatures \in [0.1, 1.0]$, and random $top\_p \in [0.7, 1.0]$.

	Prediction performance (MAE)	Inference throughput (samples/s)
Fixed hyperparameters	119.2315	0.588
Varied hyperparameters	131.0613	0.554

performs Zheng's baseline method by 29.51% on LLaMA2-7B and 11.49% on Qwen2.5-14B. The main reasons for this improvement are: (1) the overhead of response length prediction in our method is lower by two orders of magnitude; (2) our clustering-based grouping strategy, which groups queries according to the response length range, is more effective than directly grouping by maximum length.

6 Generalization discussion

In this section, we investigate the generalization ability of the proposed method. For all experiments herein, LLaMA2-7B is employed as the default base model. The discussion aims to answer the following questions.

- RQ1: How does the model generalize under different hyperparameter settings?
- RQ2: Can a model trained on one dataset generalize to another?
- RQ3: What is the trend of predictive performance with varying training data scales?
- RQ4: If the model is trained on a mixture of data from several heterogeneous domains, is it able to maintain high performance across the corresponding test sets?

6.1 Hyperparameters sensitivity (RQ1)

Intuitively, decoding hyperparameters such as *temperature* and *top_p* can significantly affect the output length of language models. To investigate their impact, we randomly selected a set of queries and conducted controlled experiments. For each query, we generated 100 responses under each combination of *temperature* and *top_p* settings. We then plotted the response length distributions using histograms and kernel density estimation (KDE). Representative examples are shown in Figure 9. From these results, we can draw an important conclusion: although temperature and *top_p* strongly influence the response of individual queries, their effect on the overall length distribution is relatively minor. While higher *temperature* increases response variability and lower *temperature* reduces it, the overall trends in length distribution remain relatively stable across different hyperparameter settings.

This finding suggests that our length prediction model may generalize well across different decoding hyperparameters. To further validate this, we trained our model with fixed hyperparameters ($temperature = 0.6$, $top\_p = 0.9$), and evaluated it under randomly varying settings (temperature $\in [0.1, 1.0]$, $top\_p \in [0.7, 1.0]$). As shown in Table 5, our model exhibited only a modest degradation in MAE (9.92%). The inference throughput saw a minor decrease from 0.588 to 0.554 samples/s compared with fixed hyperparameters, while it still significantly outperformed the vanilla method's baseline (0.413 samples/s, see Table 4).

Finally, as previously discussed, higher *temperature* increases response variability and lower *temperature* reduces it. This behavior may potentially be further corrected through the temperature scaling method [43], which we leave as future work.

6.2 Generalization across datasets (RQ2)

To evaluate the generalization capability of our method on unseen queries and different prompting paradigms, we conducted an experiment to assess whether a model trained on one dataset can generalize to another. Specifically, we employed the GSM-8K and the Instruction-in-Wild dataset. In the GSM-8K dataset, each query is structured with embedded reasoning examples, following a zero-shot in-context learning framework.

As shown in Table 6, our model achieves a 21.25% lower MAE than the point estimation method on the Instruction-in-Wild dataset (169.33 vs. 215.03), and its performance is also comparable to that of a model directly trained on Instruction-in-Wild (MAE of 163.13, as reported in Table 3). Moreover, based on the predicted lengths, our sequence scheduling method achieves a 37.32% improvement on throughput over the vanilla baseline. For the GSM-8K dataset under the context-learning setting, our model reduces the

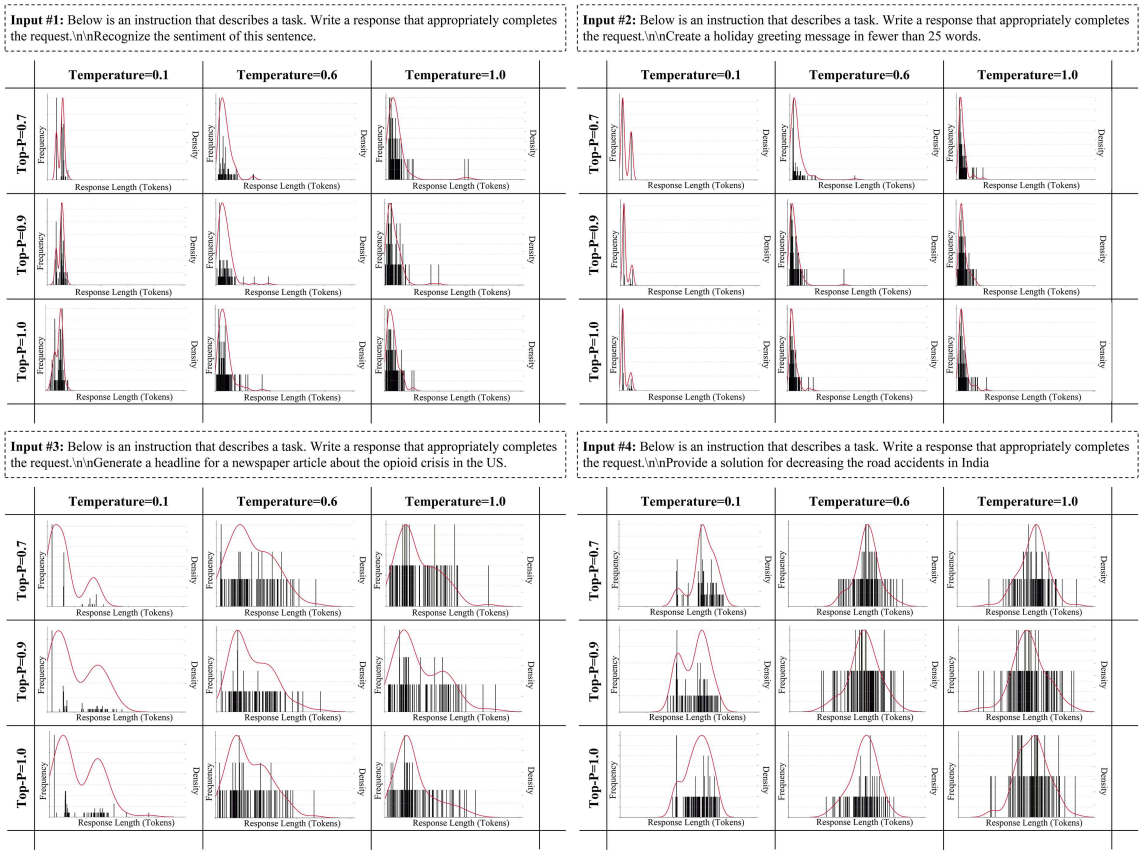

Figure 9 (Color online) Response length distributions using histograms and kernel density estimation under different hyperparameter settings.

Table 6 Performance of model trained on the Alpaca dataset and evaluated on Instruction-in-Wild and GSM-8K.

	Prediction performance (MAE)		Inference throughput (samples/s)		
	Our method	Point estimation	Our method	Vanilla	Improvement
Instruction-in-wild	169.3260	215.0257	0.447	0.325	**37.32%**
GSM-8K	454.1903	716.0652	0.342	0.297	**15.10%**

length prediction MAE by 36.57% compared with the point estimation method (454.19 vs. 716.07), yet the absolute error remains relatively high (see Table 7, the MAE can be reduced to 192.48 when trained directly on GSM-8K). The reason is that GSM-8K focuses on mathematical reasoning, which differs in domain from Instruction-in-Wild and Alpaca. Furthermore, in the GSM-8K dataset, a substantial number of responses reach the maximum output length limit (1024 tokens), while our predictor was trained on data with very limited examples exceeding this threshold. Despite this challenge, our method still achieves a 15.10% throughput improvement over the vanilla baseline, because even coarse predictions can still provide some useful signals for scheduling. However, optimal performance is expected when the model has previously been exposed to similar input types.

6.3 Performance with varying training data scales (RQ3)

As analyzed in Subsection 6.2, optimal performance is expected when the model has previously been exposed to similar input types. This, however, may raise a new concern: the model's potential reliance on very large-scale datasets. To address this concern, we investigated how predictive performance scales with the amount of training data. For both datasets, we trained the model using incrementally sampled subsets of the training data (5%, 10%, 15%, 25%, 50%, 75%, and 100%) and evaluated four key metrics: mean absolute error (MAE), mean interval score (MIS), interval score (IS), and expected calibration error (ECE). Figure 10(a) illustrates the experimental results. For the Alpaca dataset, the model achieved near-optimal performance with approximately 50% of the training data. This indicates that our length

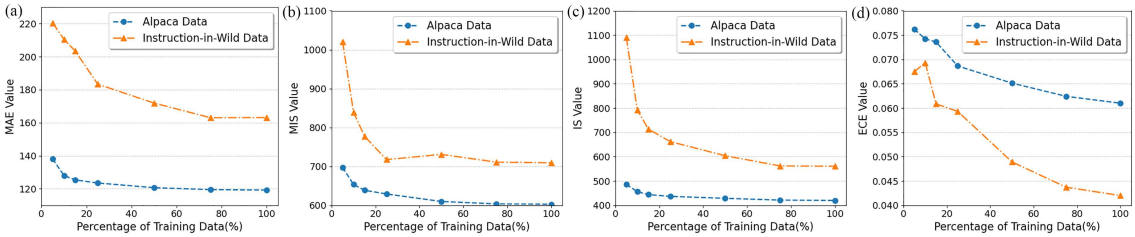

Figure 10 (Color online) How predictive performance (a) MAE, (b) MIS, (c) IS, and (d) ECE scales with the amount of training data.

Table 7 Performance of model trained on the Alpaca dataset and evaluated on Instruction-in-Wild and GSM-8K.

	Prediction performance (MAE)		Inference throughput (samples/s)		
	Trained on mixture dataset	Trained on single dataset	Our method	Vanilla	Improvement (%)
Alpaca	120.8307	119.2315	0.577	0.413	**39.71**
Instruction-in-wild	159.5449	163.1294	0.458	0.325	**40.92**
GSM-8K	195.3575	192.4831	0.402	0.297	**35.35**

prediction model can effectively capture the relationship between input queries and output lengths from moderate data scales. In contrast, the Instruction-in-Wild dataset required a larger proportion of training data to reach comparable performance levels. Specifically, the model needed nearly 75% of the training data to achieve its near-optimal predictive capability. This slower convergence is due to the limited size of the data set, only 429 queries. Overall, these results confirm that our method can efficiently capture latent input-output length correlations from limited data, avoiding overreliance on large-scale datasets.

6.4 Generalization across heterogeneous domains (RQ4)

As shown in the experiments of Subsection 6.2, when the model encounters queries from previously unseen domains, its prediction performance tends to degrade. In this subsection, we investigate whether a model trained on a mixture of data from several heterogeneous domains can maintain high performance across the corresponding test sets. Specifically, we selected the Alpaca, Instruction-in-Wild, and GSM-8K dataset. In the GSM-8K dataset, each query is structured with embedded reasoning examples, following a zero-shot in-context learning framework. We randomly sampled 50% of the training data from each dataset to construct a mixture dataset for training. The results, shown in Table 7, indicate that the length prediction performance of the model trained on the mixture dataset is comparable to that of models trained individually on each dataset. Furthermore, based on predicted lengths, the scheduling method achieves a significant throughput improvement upon all three datasets. These results demonstrate that when trained on a dataset representative of multiple domains, the model can generalize well across them.

7 Related study

Efficient LLM inference. LLMs have become an important tool for knowledge-based industries and provide support for many human-centric tasks due to their powerful reasoning ability in language [44,45]. However, it usually requires higher computational costs and usage rates to deal with the large number of requests. Achieving efficient inference for LLMs has undoubtedly become an important way to effective deployment. The Parallelism method includes data parallelism [46], tensor parallelism [47] and pipeline parallelism [48], whose core is to divide the corresponding part into multiple groups and allocate them onto multiple GPUs for parallel computing, thereby reducing latency and improving device throughput, alleviating the problem that LLM parameters may not be able to be stored in a single computing device due to their large size. Quantization reduces the amount of memory occupied and accessed by converting the weight and activation of a model from high-precision to low-precision representation, such as Refs. [49–51] concentrate on weight quantization, Refs. [52,53] focus on both weight and activation, and Refs. [54, 55] focus on compression of key value (KV) caching. Additionally, there is still some work focused on optimizing LLM batch processing, Refs. [56, 57] adopt a simple FCFS strategy to meet different scheduling requests. Deepspeed-fastgen [58] utilizes continuous batch processing and non-continuous KV

caching technology to improve the hardware utilization rate and response speed of LLMs. FastServe [12] proposes a preemptive scheduling strategy to optimize the head-of-line blocking problem and achieve lower job completion time (JCT). Cheng et al. [59] proposed a batch prompting strategy for few-shot in-context learning to reduce tokens and inference costs. Our method focuses on the uncertainty in response length perception tasks and provides more reliable length prediction intervals. Finally, we can provide better batching strategies based on these prediction results.

Uncertainty quantification. In many key areas such as healthcare [60] and autonomous driving [61], uncertainty quantification can help models know "when they don't know", and also help humans identify when not to trust model predictions, thereby reducing potential risks. Therefore, the study of uncertainty quantification in deep learning has received increasing interest in recent years. Gal et al. [30] treated the prediction with dropout as an approximate Bayesian inference in deep Gaussian processes. Bayesian neural networks [62] use probability distributions instead of single values to represent weights, which can provide uncertainty information while giving a prediction. However, both methods require significant computational and memory overhead, which severely hinders their deployment. Unlike that, evidential deep learning directly places prior information on the likelihood function and trains neural networks to output hyperparameters of higher-order evidence distributions to represent uncertainty in discrete classification [31, 63, 64] and continuous regression problems [32, 37]. In this study, we are the first to discuss the uncertainty quantification methods in the response length perception task, and this new paradigm will yield a more reliable model result within a confidence interval for length predictions. Finally, we also benchmark the four uncertainty quantification methods on two public prompt datasets.

Response length perception. Due to the lack of a clear purpose, there is not much work being applied to the response length perception tasks. At present, most works focus on the non-autoregressive translation tasks [65] because their parallel generation mechanism requires prior knowledge of the approximate length of the output sequence in order to allocate resources to each part of the sequence. For example, Sun et al. [66] obtained the target sentence length based on the sentence lengths in the source dataset, a constant bias term calculated from the train data, and Gu et al. [67] proposed to predict the overall sentence length by predicting the number of tokens converted from each input token with the fertility model [68]. Several methods, such as [69, 70], train the model to predict the length of a sequence as a response length output by adding a special token *LENGTH* to the encoder, and methods [71, 72] pool the outputs of the encoder into a length classifier. In addition, Zheng et al. [16] had attempted to improve their length prediction capabilities through modifying prompts and fine-tuning methods. However, current prediction methods are all simple point estimation and do not consider the uncertainty of length itself.

8 Conclusion

In this paper, we discuss why uncertainty quantification is important in LLM response length perception tasks. We then propose a new paradigm that incorporates uncertainty quantification, allowing the model to generate confidence intervals for its predictions. We further benchmark four uncertainty quantification methods, including both Frequentist and Bayesian approaches, and find that the EDL is the most effective and efficient for this task. Moreover, we conducted a case study to evaluate whether the predicted response length could guide the batching strategy and thereby improve LLM inference efficiency. The results demonstrate that our proposed uncertainty-aware approach averagely reduces the inference time by 38.14% compared with random batching and by 20.50% compared with the state-of-the-art approach without uncertainty quantification [16].

Compared with traditional methods, our approach provides confidence intervals for predictions (i.e., ranges for the predicted response lengths). However, there is still room for improvement. If we can also obtain probability density distributions for each predicted response length, the optimal batching strategy can be calculated directly. We plan to explore this in future work.

Acknowledgements This work was supported by Key Research and Development Project in Shaanxi Province (Grant No. 2023GXLH-024), National Natural Science Foundation of China (Grant Nos. 62476215, 62302380, 62037001, 62137002, 62192781), and China Postdoctoral Science Foundation (Grant No. 2023M742789). We acknowledge Kaihao ZHANG, Zheng ZHAO, and Hao WU for their contributions.

References

1 Zhao W X, Zhou K, Li J, et al. A survey of large language models. ArXiv:2303.18223

2 Chang Y, Wang X, Wang J, et al. A survey on evaluation of large language models. ACM Trans Intell Syst Tech, 2024, 15: 1–45
3 Lyu C, Xu J, Wang L. New trends in machine translation using large language models: case examples with ChatGPT. ArXiv:2305.01181
4 Zhong L, Wang Z. Can LLM replace stack overflow? A study on robustness and reliability of large language model code generation. In: Proceedings of the AAAI Conference on Artificial Intelligence, 2024. 21841–21849
5 Li N, Gao C, Li Y, et al. Large language model-empowered agents for simulating macroeconomic activities. ArXiv:2310.10436
6 Orenstrakh M S, Karnalim O, Suarez C A, et al. Detecting LLM-generated text in computing education: a comparative study for ChatGPT cases. ArXiv:2307.07411
7 Cascella M, Montomoli J, Bellini V, et al. Evaluating the feasibility of ChatGPT in healthcare: an analysis of multiple clinical and research scenarios. J Med Syst, 2023, 47: 33
8 Fei Z, Shen X, Zhu D, et al. Lawbench: benchmarking legal knowledge of large language models. ArXiv:2309.16289
9 Aminabadi R Y, Rajbhandari S, Awan A A, et al. Deepspeed- inference: enabling efficient inference of transformer models at unprecedented scale. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2022. 1–15
10 Crankshaw D, Wang X, Zhou G, et al. Clipper: a low-Latency online prediction serving system. In: Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation, 2017. 613–627
11 Qiu H, Mao W, Patke A, et al. Efficient interactive LLM serving with proxy model-based sequence length prediction. ArXiv:2404.08509
12 Wu B, Zhong Y, Zhang Z, et al. Fast distributed inference serving for large language models. ArXiv:2305.05920
13 Prekas G, Kogias M, Bugnion E. Zygos: achieving low tail latency for microsecond-scale networked tasks. In: Proceedings of the 26th Symposium on Operating Systems Principles, 2017. 325–341
14 Gurevich P, Stuke H. Gradient conjugate priors and multi-layer neural networks. Artif Intell, 2020, 278: 103184
15 Radford A, Wu J, Child R, et al. Language models are unsupervised multitask learners. OpenAI blog, 2019, 1: 9
16 Zheng Z, Ren X, Xue F, et al. Response length perception and sequence scheduling: an LLM-empowered LLM inference pipeline. Adv Neural Inf Process Syst, 2024, 36: 65517–65530
17 Olston C, Fiedel N, Gorovoy K, et al. TensorFlow-serving: flexible, high-performance ML serving. ArXiv:1712.06139
18 Taori R, Gulrajani I, Zhang T, et al. Stanford Alpaca: an instruction-following LLAMA model. 2023. https://github.com/tatsu-lab/stanford_alpaca
19 He K, Zhang X, Ren S, et al. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016. 770–778
20 Tolstikhin I O, Houlsby N, Kolesnikov A, et al. MLP-mixer: an all-MLP architecture for vision. Adv Neural Inf Process Syst, 2021, 34: 24261–24272
21 Gujarati A, Karimi R, Alzayat S, et al. Serving DNNs like clockwork: performance predictability from the bottom up. In: Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation, 2020. 443–462
22 Wei J, Tay Y, Bommasani R, et al. Emergent abilities of large language models. ArXiv:2206.07682
23 Wei J, Wang X, Schuurmans D, et al. Chain-of-thought prompting elicits reasoning in large language models. Adv Neural Inf Process Syst, 2022, 35: 24824–24837
24 Nye M, Andreassen A J, Gur-Ari G, et al. Show your work: scratchpads for intermediate computation with language models. ArXiv:2112.00114
25 Hu E J, Shen Y, Wallis P, et al. Lora: low-rank adaptation of large language models. ArXiv:2106.09685
26 Fu Z, Lam W, Yu Q, et al. Decoder-only or encoder-decoder? Interpreting language model as a regularized encoder-decoder. ArXiv:2304.04052
27 Zhang X, Li Z, Zhang Y, et al. Language models are universal embedders. ArXiv:2310.08232
28 Wu D, Gao L, Chinazzi M, et al. Quantifying uncertainty in deep spatiotemporal forecasting. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021. 1841–1851
29 Koenker R, Hallock K F. Quantile regression. J Econ Perspect, 2001, 15: 143–156
30 Gal Y, Ghahramani Z. Dropout as a Bayesian approximation: representing model uncertainty in deep learning. In: Proceedings of the 33rd International Conference on Machine Learning, 2016. 1050–1059
31 Sensoy M, Kaplan L, Kandemir M. Evidential deep learning to quantify classification uncertainty. In: Advances in Neural Information Processing Systems, 2018
32 Amini A, Schwarting W, Soleimany A, et al. Deep evidential regression. Adv Neural Inf Process Syst, 2020, 33: 14927–14937
33 Parisi G, Shankar R. Statistical field theory. Phys Today, 1988, 41: 110
34 Xue F, Jain K, Shah M H, et al. Instruction in the wild: a user-based instruction dataset. 2023. https://github.com/XueFuzhao/InstructionWild
35 Touvron H, Martin L, Stone K, et al. LLAMA 2: open foundation and fine-tuned chat models. ArXiv:2307.09288
36 Yang A, Yang B, Zhang B, et al. Qwen2.5 technical report. ArXiv:2412.15115
37 Oh D, Shin B. Improving evidential deep learning via multi-task learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, 2022. 7895–7903
38 Askanazi R, Diebold F X, Schorfheide F, et al. On the comparison of interval forecasts. J Time Series Anal, 2018, 39: 953–965
39 Gneiting T, Raftery A E. Strictly proper scoring rules, prediction, and estimation. J Amer Statist Assoc, 2007, 102: 359–378
40 Guo C, Pleiss G, Sun Y, et al. On calibration of modern neural networks. In: Proceedings of the 34th International Conference on Machine Learning, 2017. 1321–1330
41 Kuleshov V, Fenner N, Ermon S. Accurate uncertainties for deep learning using calibrated regression. In: Proceedings of the 35th International Conference on Machine Learning, 2018. 2796–2804
42 Macqueen J. Some methods for classification and analysis of multivariate observations. In: Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, 1967
43 Ding Z, Han X, Liu P, et al. Local temperature scaling for probability calibration. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021. 6889–6899
44 Hoffmann J, Borgeaud S, Mensch A, et al. Training compute-optimal large language models. ArXiv:2203.15556
45 Chowdhery A, Narang S, Devlin J, et al. PALM: scaling language modeling with pathways. J Mach Learn Res, 2023, 24: 1–113
46 Gholami A, Azad A, Keutzer K, et al. Integrated model and data parallelism in training neural networks. ArXiv:1712.04432
47 Li S, Liu H, Bian Z, et al. Colossal-AI: a unified deep learning system for large-scale parallel training. In: Proceedings of the 52nd International Conference on Parallel Processing, 2023. 766–775
48 Huang Y, Cheng Y, Bapna A, et al. GPipe: efficient training of giant neural networks using pipeline parallelism. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems, 2019. 103–112
49 Frantar E, Ashkboos S, Hoefler T, et al. GptQ: accurate post-training compression for generative pretrained transformers. ArXiv:2210.17323
50 Park G, Park B, Kim M, et al. Lut-gemm: quantized matrix multiplication based on LUTs for efficient inference in large-scale

generative language models. ArXiv:2206.09557
51 Lin J, Tang J, Tang H, et al. AWQ: activation-aware weight quantization for LLM compression and acceleration. ArXiv:2306.00978
52 Dettmers T, Lewis M, Belkada Y, et al. LLM. int8 (): 8-bit matrix multiplication for transformers at scale. ArXiv:2208.07339
53 Xiao G, Lin J, Seznec M, et al. Smoothquant: accurate and efficient post-training quantization for large language models. In: Proceedings of the 40th International Conference on Machine Learning, 2023. 38087–38099
54 Ribar L, Chelombiev I, Hudlass-Galley L, et al. SparQ attention: bandwidth-efficient LLM inference. ArXiv:2312.04985
55 Kang H, Zhang Q, Kundu S, et al. Gear: an efficient KV cache compression recipe for near-lossless generative inference of LLM. ArXiv:2403.05527
56 Kwon W, Li Z, Zhuang S, et al. Efficient memory management for large language model serving with paged attention. In: Proceedings of the 29th Symposium on Operating Systems Principles, 2023. 611–626
57 Yu G I, Jeong J S, Kim G W, et al. Orca: a distributed serving system for Transformer-based generative models. In: Proceedings of the 16th USENIX Symposium on Operating Systems Design and Implementation, 2022. 521–538
58 Holmes C, Tanaka M, Wyatt M, et al. DeepSpeed-Fastgen: high-throughput text generation for LLMs via MII and DeepSpeed-inference. ArXiv:2401.08671
59 Cheng Z, Kasai J, Yu T. Batch prompting: efficient inference with large language model APIs. ArXiv:2301.08721
60 Chua M, Kim D, Choi J, et al. Tackling prediction uncertainty in machine learning for healthcare. Nat Biomed Eng, 2023, 7: 711–718
61 Michelmore R, Kwiatkowska M, Gal Y. Evaluating uncertainty quantification in end-to-end autonomous driving control. ArXiv:1811.06817
62 Blundell C, Cornebise J, Kavukcuoglu K, et al. Weight uncertainty in neural network. In: Proceedings of the 32nd International Conference on Machine Learning, 2015. 1613–1622
63 Joo T, Chung U, Seo M G. Being Bayesian about categorical probability. In: Proceedings of the 37th International Conference on Machine Learning, 2020. 4950–4961
64 Malinin A, Gales M. Predictive uncertainty estimation via prior networks. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, 2018
65 Gu J, Tan X. Non-autoregressive sequence generation. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts, 2022. 21–27
66 Sun Z, Li Z, Wang H, et al. Fast structured decoding for sequence models. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems, 2019. 3016–3026
67 Gu J, Bradbury J, Xiong C, et al. Non-autoregressive neural machine translation. ArXiv:1711.02281
68 Brown P F, Della Pietra S A, Della Pietra V J, et al. The mathematics of statistical machine translation: parameter estimation. Comput Linguist, 1993, 19: 263–311
69 Devlin J. Bert: Pre-training of deep bidirectional transformers for language understanding. ArXiv:1810.04805
70 Ghazvininejad M, Levy O, Liu Y, et al. Mask-predict: parallel decoding of conditional masked language models. ArXiv:1904.09324
71 Lee J, Mansimov E, Cho K. Deterministic non-autoregressive neural sequence modeling by iterative refinement. ArXiv:1802.06901
72 Lee J, Shu R, Cho K. Iterative refinement in the continuous space for non-autoregressive neural machine translation. ArXiv:2009.07177

Profile of Qinghua ZHENG

Prof. Qinghua ZHENG received the B.S. degree in computer software in 1990, the M.S. degree in computer organization and architecture in 1993, and the Ph.D. degree in system engineering in 1997 from Xi'an Jiaotong University. He did postdoctoral research at Harvard University from February 2002 to October 2002 and was a visiting professor at the University of Hong Kong from November 2004 to January 2005. Since 1995, he has been with the Department of Computer Science and Technology at Xi'an Jiaotong University. Currently, he is a member of the Chinese Academy of Engineering and the president of Tongji University.

Prof. Zheng is a winner of the National Funds for Distinguished Young Scientists and a Distinguished Professor for the Cheung Kong Scholar Project. He is one of the first batch of leading scientists for the "Ten-Thousand Talents Project". He is also a member of the Academic Committee on the Science and Technology Department of the Ministry of Education, and the chairman of the Steering Committee on Computer Instruction of the Ministry of Education. He is the person in charge of the Innovation Team of National Natural Science Foundation of China.

Prof. Zheng has published more than 190 papers in *IEEE TPAMI*, *TKDE*, *IJCAI*, *AAAI*, etc., and three monographs, including *Big Data Knowledge Engineering*. He hosts more than 40 scientific projects, including National High-tech R&D Program of China (863 Program), National Science and Technology Support Program, Core-electronics, High-end-general-Chips, and Infrastructural-software Project of China, the Key program of National Natural Science Foundation of China, and the Major Project of Scientific and Technological Innovation 2030 in New Generation Artificial Intelligence.

Theories, methods, and applications of big data knowledge engineering

Prof. Zheng has been engaged in long-term research in knowledge engineering. Through systematic and in-depth studies, he established the original concept and model of the Knowledge Forest. He proposed two core methods: Knowledge Forest Construction and Reasoning within Knowledge Forests. He has overcome key technical challenges such as (1) the integration of fragmented knowledge into a Knowledge Forest and (2) the generation of reasoning chains using Knowledge Forest, thus forming original theoretical and methodological achievements in big data knowledge engineering. Based on these research achievements, Prof. Zheng developed the knowledge forest personalized learning system, which has seen significant applications in online education. Additionally, his work aids in the detection of tax fraud and evasion in China's Golden Tax Project, leading to nationwide deployment and helping recover substantial tax revenues. These research achievements have won several awards, including three Second Prizes of the State Science and Technology Progress Awards, two First Prizes of National Teaching Achievement Award, three Second Prizes of National Teaching Achievement Award, five Ministerial and Provincial-Level Science and Technology Progress Awards, one Science and Technology Progress Award of Ho Leung Ho Lee Foundation (2022), one Outstanding Science and Technology Progress Award of Chinese Association of Automation.

Selected publications

• Zheng Q H, Zhang L L, Gong T L, et al. Big Data Knowledge Engineering. Beijing: Science Press, 2023

• Zheng Q H, Liu J, Wei B F, et al. Knowledge Forest: Theory, Methods, and Practice. Beijing: Science Press, 2021

• Zheng Q H, Liu J, Tian F, et al. Web Knowledge Mining: Theory, Methods, and Applications. Beijing: Science Press, 2010

• Zheng Q H, Cao S Z, Ruan J F, et al. Label-noise learning via mixture proportion estimation (in Chinese). Sci Sin Inform, 2024, 54: 603–622

• Zheng Q H, Shi B, Dong B. Technologies and applications of big data knowledge engineering for smart taxation systems. Strategic Study Chin Acad Eng, 2023, 25: 221–231

• Zheng Q H, Liu J, Zeng H W, et al. Knowledge forest: a novel model to organize knowledge fragments. Sci China Inf Sci, 2021, 64: 179103

• Zheng Q H, Peng Z, Dang Z H, et al. Deep tabular data modeling with dual-route structure-adaptive graph networks. IEEE Trans Knowl Data Eng, 2023, 35: 9715–9727

• Zheng Q H, Xu Y M, Liu H X, et al. A survey of tax risk detection using data mining techniques. Engineering, 2024, 34: 43–59