

# Look Before You Leap: An Exploratory Study of Uncertainty Analysis for Large Language Models

Yuheng Huang, Jiayang Song, Zhijie Wang, Shengming Zhao, Huaming Chen, Felix Juefei-Xu, and Lei Ma<sup>✉</sup>

**Abstract**—The recent performance leap of Large Language Models (LLMs) opens up new opportunities across numerous industrial applications and domains. However, the potential erroneous behavior (e.g., the generation of misinformation and hallucination) has also raised severe concerns for the trustworthiness of LLMs, especially in safety-, security- and reliability-sensitive industrial scenarios, potentially hindering real-world adoptions. While uncertainty estimation has shown its potential for interpreting the prediction risks made by classic machine learning (ML) models, the unique characteristics of recent LLMs (e.g., adopting self-attention mechanism as its core, very large-scale model size, often used in generative contexts) pose new challenges for the behavior analysis of LLMs. Up to the present, little progress has been made to better understand whether and to what extent uncertainty estimation can help characterize the capability boundary of an LLM, to counteract its undesired behavior, which is considered to be of great importance with the potential wide-range applications of LLMs across industry domains. To bridge the gap, in this paper, we initiate an early exploratory study of the risk assessment of LLMs from the lens of uncertainty. In particular, we conduct a large-scale study with as many as twelve uncertainty estimation methods and eight general LLMs on four NLP tasks and seven programming-capable LLMs on two code generation tasks to investigate to what extent uncertainty estimation techniques could help characterize the prediction risks of LLMs. Our findings confirm the potential of uncertainty estimation for revealing LLMs’ uncertain/non-factual predictions. The insights derived from our study can pave the way for more advanced analysis and research on LLMs, ultimately aiming at enhancing their trustworthiness.

**Index Terms**—Large Language Models, Deep Neural Networks, Uncertainty Estimation, Software Reliability

## I. INTRODUCTION

**L**ARGE Language Models (LLMs) have demonstrated impressive capabilities in miscellaneous Natural Language Processing (NLP) tasks and promising adaptability in practical applications across diverse domains, including but not limited to content moderation [1], code generation [2], conversational AI [3], and personalized content recommendations [4]. The scale of deployment is vast, addressing the needs of diverse user demographics and industries. As a prominent example,

Yuheng Huang is with The University of Tokyo, Tokyo 113-8658, Japan (e-mail: yuhenghuang42@g.ecc.u-tokyo.ac.jp).

Jiayang Song, Zhijie Wang and Shengming Zhao are with the University of Alberta, Edmonton, T6G 1H9, Canada (e-mail: {jiayan13, zhijie.wang, shengmi1}@ualberta.ca).

Huaming Chen is with The University of Sydney, Australia (e-mail: huaming.chen@sydney.edu.au).

Felix Juefei-Xu is with New York University, New York, NY 10012, USA (e-mail: juefei.xu@nyu.edu).

Lei Ma is with The University of Tokyo, Tokyo 113-8658, Japan, and also with the University of Alberta, Canada (e-mail: ma.lei@acm.org).

<sup>✉</sup> Lei Ma is the corresponding author.

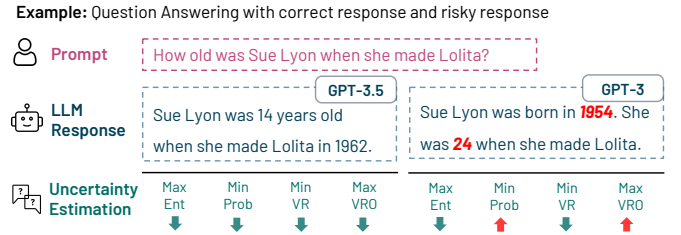


Fig. 1: Uncertainty estimation for a QA task.

Meta has launched foundation models such as the Llama family [5], [6]. By September 2023, these models had driven the creation of over 3,500 enterprise projects and inspired more than 7,000 GitHub repositories [7].

Despite the attractive performance that LLMs present and their rapid evolution within both academics and industries, an urgent common concern about LLMs has the propensity of generating erroneous information without warning. Such phenomenon of erroneous generation can exhibit in terms of different manifestations (e.g., hallucination [8], disinformation [9], bias [10]) across various tasks. In general, the current LLMs are found to have the tendency to generate problematic, nonfactual responses that are not from training sources or misguided by biased data. However, these responses are often presented in a natural human-like tone [11], [12]. Such characteristics cause erroneous information to be highly mixed and intertwined with confident and factual contexts, making their detection and localization difficult without close inspection and diligent fact-checking [13]. As an example, Fig. 1 (GPT-3) depicts an example of an LLM answering a question with nonfactual information.

Risk assessments thus become crucial in the process of mitigating such threats. A recent survey highlights that 98% of respondents, encompassing domain experts and civil society members, firmly believe that AGI (artificial general intelligence) labs should undertake risk assessments before deployment [14]. For the AI industry, implementing comprehensive risk assessment methods is not just a technical necessity but also an ethical obligation. Major tech corporations [15] such as Microsoft [16], OpenAI [17], Amazon [18], and Google [19], along with non-governmental organizations (NGOs, e.g., the Centre for the Governance of AI [20]), are fervently working towards developing safe, secure, transparent, reliable and responsible LLMs and AGI applications. As a driving force behind open and collaborative AI research, Meta also commits substantial resources to the development of responsible AI, emphasizing trustworthiness, transparency,

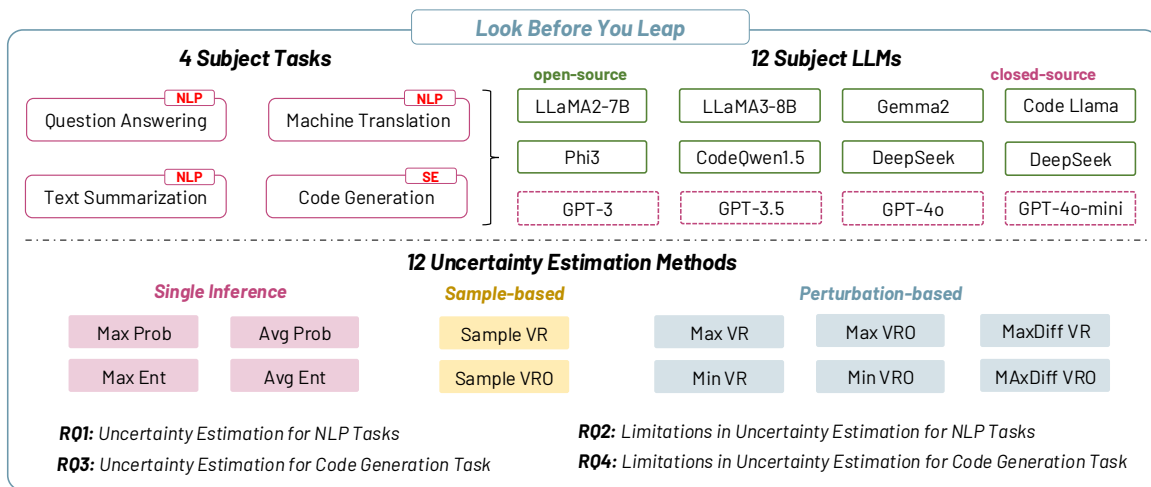


Fig. 2: A high-level overview and workflow of this paper.

robustness, etc. [21], [22]. These endeavors encompass a range of reports (e.g., *Building Generative AI Responsibly* [23]), open-source tools (e.g., the model interpretability framework *Captum* [24]), and datasets (e.g., *Hateful Memes* [25]).

Although there has been substantial work on other AI models, risk assessments for LLMs are still in their infancy. Due to their billions of parameters, vast amounts of (often inaccessible) data, and potential closed-source nature, LLMs present significant challenges for analysis and safeguarding. Yet, such a safeguard is crucial, especially considering the widespread adoption of LLMs. Stakeholders from research communities, industry, open source initiatives, NGOs, and businesses may all be negatively affected by untrustworthy LLMs. Uncertainty estimation, aimed at gauging the confidence level of model outputs [26], [27], [28], stands out as a promising approach for identifying risks in general Machine Learning (ML) models. Such techniques also have the potential for detecting erroneous generation from LLMs [29] even under black-box settings. It is thus possible to take them as plug-and-play tools in both academic and industrial scenarios. For example, Fig. 1 shows that a higher uncertainty score could possibly indicate an erroneous generation of an LLM. However, it is still unclear whether and to what extent uncertainty estimation methods could do when measuring and characterizing an LLM’s capability limitations. Furthermore, it also raises questions such as “*Are there better practices for employing these methods in practical scenarios*”, “*Do we need further adaptations from the industrial perspective to better cater to the distinct features of LLMs (e.g., task diversity and high computational cost)*”, etc. To the best of our knowledge, up to the present, there is a lack of a general framework that integrates different uncertainty estimation methods for LLMs, as well as a systematic study to investigate the effectiveness of uncertainty estimation in characterizing an LLMs’ capabilities.

To bridge this gap, in this paper, we present an exploratory study to understand the trustworthiness of LLMs from the lens of uncertainty estimation. Considering the generality and versatility for various application scenarios, we strive to identify suitable methods to minimize the requirement of

LLMs internal information (e.g., model architecture, model parameters). Such criteria enable these methods to be seamlessly adapted and incorporated by end-users of commercial models, such as GPT-3.5 and GPT-4. Overall, we collected and implemented as many as 12 representative uncertainty estimation methods that were originally designed for general DNNs and successfully adapted them to the contexts of LLM applications. To better capture an in-depth understanding of the effectiveness of these methods, we conducted large-scale experiments with as many as twelve LLMs on both NLP (i.e., question answering, text summarization, machine translation) and software programming (i.e., code generation) tasks to analyze the correlation between uncertainty estimation results and LLMs performance. The models comprise three from *MetaAI*, four from *OpenAI*, one from *Google*, one from *Microsoft*, one from *Alibaba*, one from *DeepSeek*, and one from *BigCode*. We also evaluated five older LLMs (e.g., GPT-2, LLaMA, Codegen, Incoder, and Santacoder), with the corresponding results available on our website. In total, we have evaluated 17 LLMs. The overall workflow of our work is shown in Fig. 2. In particular, we investigate the following research questions:

- **RQ1:** To what extent can the uncertainty estimation techniques help identify potential risks of LLMs in NLP tasks?
- **RQ2:** What limitations do the uncertainty estimation methods encounter when applied to LLMs in the context of NLP tasks?
- **RQ3:** To what extent can the uncertainty estimation methods assist in identifying potential risks of LLMs for code generation?
- **RQ4:** What potential limitation do the uncertainty estimation methods face when being applied to LLMs for code generation?

Our findings validate that uncertainty measurement can, to an extent, be helpful in detecting erroneous responses in general NLP tasks. Additionally, it has also shown to be promising as an indicator for pinpointing faulty programs produced by LLMs. Even though, these methods might fall short in detecting nuanced errors made by high-performance commercial models. They seem better suited for filtering out

more blatant mistakes. Based on the results, we further discuss the insights from our study and highlight a few potential research directions of leveraging uncertainty estimation to enhance the trustworthiness of LLMs for real-world applications across domains. *First*, research efforts are needed with novel uncertainty estimation techniques exclusively for LLMs to better fit the corresponding diverse task-handling ability. *Second*, we observe that different LLMs can sometimes display markedly distinct uncertain behaviors. Consequently, even though these methods are inherently model-agnostic, stakeholders may need to undertake model-specific optimizations to achieve enhanced performance. *Furthermore*, we observe that the prompt template used in the reinforcement learning from human feedback (RLHF [30]) could potentially impact the accuracy of uncertainty estimation.

The contributions of this paper are summarized as follows:

- We collected and implemented *twelve* different uncertainty estimation methods that are successfully adapted to enable the analysis of LLM, which are also applicable to both open-source and closed-source LLM models across different downstream tasks in the grey-box setting.
- We conducted a large-scale evaluation with nine LLMs on *six* tasks from *four* different domains.
- We provided an in-depth analysis of the challenges in existing uncertainty methods for LLMs and distilled a set of implications and future opportunities toward reliable and trustworthy LLMs.
- Our toolkit, encompassing the dataset, LLM inference, and uncertainty measurement protocols, will be made available for future research endeavors.

### The Contributions to the Software Engineering Field.

LLMs have revolutionized various aspects of software engineering [31], [32], including but not limited to automated code generation [33], [2], [34], [35], software testing [36], [37], [38], [39], debugging [40], program repair [41], [42], and document generation [43]. While LLMs can serve as a critical core for many new-era AI-enabled intelligent systems in the software engineering domain, their black-box nature and inherent uncertainties pose challenges for them to be applied in the real world in a transparent, reliable, safe, and secure way. It is thus urgent to investigate and explore effective quality assurance methods. Measuring uncertainty and taking it as an indicator of AI models' reliability has been studied extensively in the SE community [44], [45], [46]. While promising, most of them focus on classification tasks with relatively simple neural architectures. On the contrary, we initialize a very early stage study on autoregressive, large-scale language models and perform various uncertainty measurements across a wide spectrum of tasks.

We further provide more supplementary results and details as well as the source code to reproduce our study at our website: <https://sites.google.com/view/llm-uncertainty>.

## II. BACKGROUND AND RELATED WORK

### A. Large Language Models

In general, a *language model* models sequences of words as a probability distribution, which can be further used to

generate coherent and contextually relevant text via conditioning based on a given prompt. Representative traditional language models include HMM (hidden Markov model) [48], *n*-gram [49], and RNN (recurrent neural networks) [50]. Recently, a specific type of neural network architecture, i.e., Transformer [51], has achieved attractive performance on language modelling. Large language models (LLMs) now typically refer to those Transformer-based language models pre-trained with large-scale text corpus and billions of parameters. LLMs have also achieved promising performance in many downstream tasks, e.g., text classification [52], text summarization [53], and machine translation [54].

Based on different Transformer architectures and pre-training tasks, LLMs largely fall into three categories: *encoder-only*, *encoder-decoder*, and *decoder-only*. *Encoder-only* LLMs also refer to masked language models (MLM), which are pre-trained through masking a certain portion of tokens (e.g., 15%) in a sequence. The training objective is to predict those masked tokens correctly. Representative *encoder-only* LLMs include BERT [55], RoBERTa [56], GraphCodeBERT [57], etc. Different from *encode-only* LLMs, *encoder-decoder* LLMs, such as BART [58], CodeT5 [59], are pre-trained through masked span prediction (MSP). *Encoder-decoder* LLMs first learn a representation from input prompts before decoding into another sequence. They are thus usually trained for sequence-to-sequence purposes. Lately, *decoder-only* LLMs have become the mainstream of LLMs research due to their training efficiency and scalability for large-scale datasets and complex model architectures (i.e., billions of model parameters). *Decoder-only* LLMs are autoregressive models. Their training objective is to predict the next token given all previous (left-only) tokens. GPT-based (generative pre-trained transformers) models (e.g., GPT2 [60], GPT3 [47], LLaMA [5] and LLaMA-2 [6]) all belong to this category. In this work, we mainly focus on *decoder-only* LLMs since they have SOTA performance. We further detail subject LLMs in our study in Sec. IV-A.

Though LLMs are pre-trained without specific tasks in mind, they can often be used for downstream tasks in two ways, i.e., through (1) prompting and (2) fine-tuning. Prompting refers to the process of in-context learning that “teaches” an LLM to solve a specific task by injecting certain knowledge and instructions into the input prompts. Different from prompting, fine-tuning is the process of updating an LLM's neural network weights through a supervised learning strategy for certain tasks (e.g., text classification). Recently, *decoder-only* LLMs have also been used with reinforcement learning from human feedback (RLHF) to improve their performance in understanding complex input prompts and following human instructions. As a result, ChatGPT (GPT3.5 with RLHF) [3] has shown superior performance in solving various complex tasks (e.g., program synthesizes [61], program repair [62]) by only following the user's instructions and feedback through a dialogue. In our study, we mainly consider LLMs through prompting (that is widely used in practice) and focus on how to estimate an LLM's uncertainty in a “black-box” way (i.e., only accessing the model's prediction output probabilities). We introduce our experiment settings in Sec. IV-B.

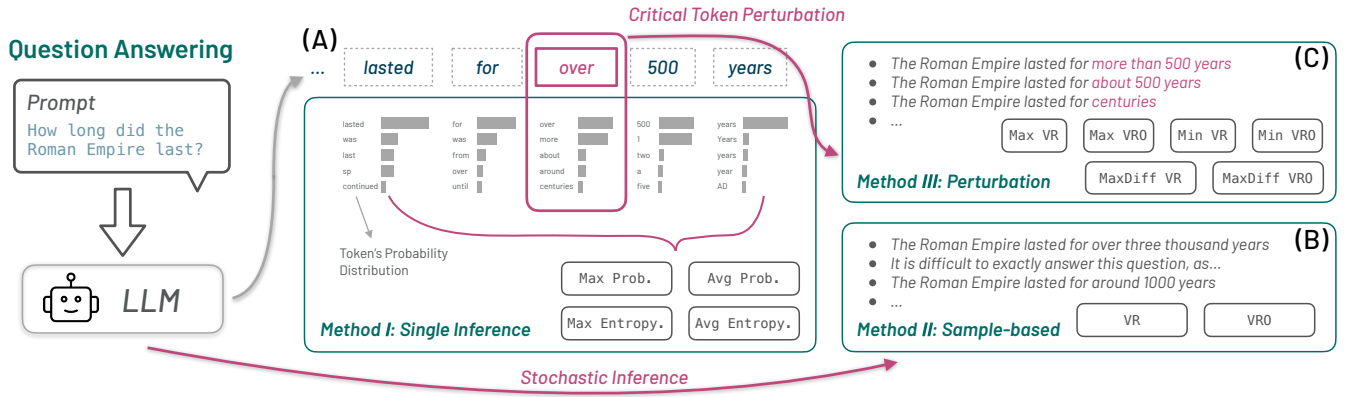


Fig. 3: A running example of how different uncertainty estimation methods work for a QA problem with GPT3 [47].

### B. Risk Assessment for ML Models

Machine learning (ML) models, especially deep learning ones, are known to be notoriously hard to interpret due to their complexity and opacity. Using ML models without appropriate risk assessment could potentially lead to particular concerns and threats regarding trustworthiness, e.g., safety [63], security [64], and ethics [65]. So far, some risk assessment techniques for general machine learning models have been proposed to reduce the impacts of these concerns [66], [67], [28], [68], [69]. Among them, there are two representative categories of risk assessment techniques: (1) data distribution analysis and (2) uncertainty estimation.

The data-driven nature of ML models requires developers to take data into account, especially when test data could be much more different compared with the training data in terms of their distribution. Data distribution analysis, including detecting distribution shift [67] and out-of-distribution samples [70], is proposed to identify such differences and avoid potential risks on unseen data. However, data distribution analysis usually requires access to training data, which is often not feasible for LLMs trained on either huge data corpus or private data corpus. Therefore, in this work, we propose to focus on *uncertainty estimation*.

In general, uncertainty estimation aims to measure an ML model's confidence level of a certain prediction. There are two main types of uncertainty in an ML model's predictions: *aleatoric* uncertainty and *epistemic* uncertainty [71]. *Aleatoric* uncertainty refers to the uncertainty that arises from observations (e.g., sensor noises in an ML model for autonomous driving). By contrast, *epistemic* uncertainty accounts for uncertainty in an ML model's parameters. Insufficient knowledge of an ML model (e.g., lack of a specific type of training data) usually leads to high *epistemic* uncertainty. In this paper, we mainly discuss estimating *epistemic* uncertainty for LLMs. Uncertainty estimation roughly falls into four categories [72]: (1) *single deterministic methods* [73], (2) *ensemble methods* [74], (3) *Bayesian methods* [75], and (4) *test-time augmentation methods* [76]. *Single deterministic methods* calculate prediction uncertainty based on one forward pass within a deterministic ML model. *Ensemble methods* estimate uncertainty based on a set of different ML models' output.

By contrast, *Bayesian methods* only leverage the inherent stochasticity of an ML model (e.g., dropout layer in deep neural networks [77]). *Test-time augmentation methods* are model-agnostic, which augment the input data at test-time to measure a model's prediction uncertainty [78]. Since we focus on the risk assessment for one standalone LLM, *ensemble methods* are excluded from our study. We detail the uncertainty estimation methods used in this work in Sec. III.

In addition to the aforementioned general risk assessment techniques, there are also a few works specified for risk assessment of LLMs [79], [80], [81], [82], [83], [84], [29]. The most related works are those proposed for uncertainty estimation of LLMs [82], [83], [84], [29]. Xiao et al. leverage *ensemble methods* to measure the natural language generation model's uncertainty and detect potential hallucinations [82]. Similarly, Malinin et al. propose a unified uncertainty estimation method for autoregressive structured prediction tasks based on *ensemble methods* [83]. To overcome the challenge of capturing "semantic equivalence" in natural language, Kuhn et al. propose *semantic entropy* that incorporates linguistic invariances created by shared meanings [84]. Recently, Manakul et al. propose SelfCheckGPT, a black-Box hallucination detection method based on token-level prediction likelihood and entropy [29]. In light of the limitations of these works, our work is the first work that is not limited to specific natural language or tasks by covering twelve uncertainty estimation methods. Furthermore, our study investigates the role of uncertainty estimation with extensive experiments with nine LLMs and six tasks, providing insights and evidence for its effectiveness as the risk assessment technique for LLMs.

## III. UNCERTAINTY ESTIMATION FOR LLMs

In this section, we first discuss the problem scenario in our study, including the corresponding assumptions. Then, we introduce our twelve uncertainty estimation techniques (three categories) based on the number of inferences required.

### A. Problem Scenario

Given an input prompt  $X = [x_1, x_2, \dots, x_n]$  ( $x_i$  denotes  $i$ th input token), an LLM  $f$  with pre-trained weights  $w$  generates another sequence  $Y = [y_1, y_2, \dots, y_m]$  ( $y_j$  denotes



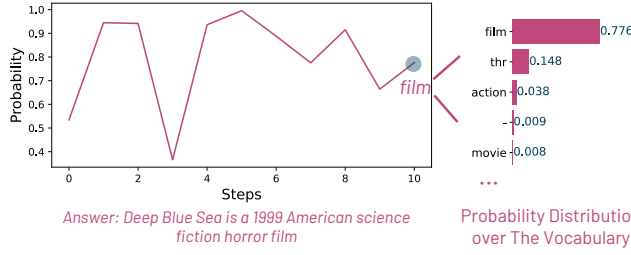


Fig. 4: The available information we can get from the LLM in our setting. This is an illustration of how *LLAMA-7B* answers the question “when did the movie deep blue sea come out?”.

$j$ th generated token) through a decoding process:  $y_j = f([X, y_1, y_2, \dots, y_{j-1}]|w)$

An uncertainty estimation method  $g$  is to calculate a score  $u$  regarding the uncertainty of  $Y$ .

Though an LLM can be regarded as an ML model, it is limited by the inherent properties of some existing uncertainty estimation methods. Following we discuss the unique characteristics of LLMs and challenges in uncertainty estimation for LLMs compared with other ML models.

- **Complexity.** The state-of-the-art (SOTA) LLMs are usually pre-trained with billions of parameters (e.g., GPT-3 [47] model has 96 layers with 6.7 billion parameters). Therefore, “white-box” analysis for interpreting LLMs (e.g., inspecting neuron activation [85], inspecting attention values [86], [87]) that requires both significant manual and computational efforts is not feasible.
- **Opacity.** There is also a lack of opacity in SOTA LLMs. First, the SOTA LLMs are usually trained with large-scale text corpus, where such data can be either publicly available or from private sources. Therefore, risk assessment techniques that require access to training data (e.g., OOD detection) can not be used in our context. Additionally, some of the SOTA LLMs are potentially proprietary assets for a company (e.g., GPT-3 [47]), where one can only access the inference results through provided APIs.
- **Task diversity.** Though the usage of LLMs can be described in a general decoding form, tasks that LLMs can solve are of greater diversity. Notably, LLMs can be used for user-defined tasks through prompting/few-shot learning (Sec. II-A). Therefore, uncertainty estimation methods that are proposed for a specific narrow domain (e.g., text classification) are hard to be used as a general risk assessment technique for LLMs.

With these characteristics and challenges in mind, we show the information that we can get from the majority of both open-source and closed-source LLMs in Fig. 3 (A) and explore possible existing solutions in this grey box setting for a general usage risk assessment. We summarize the necessary information needed in this study in Fig. 4. In most cases, we are only able to obtain the output tokens as well as the probability distribution over the vocabulary for each token. Specifically, for closed-source models such as GPT-3 [47] and GPT-3.5 [88], one is only able to obtain the top- $k$  probabilities (i.e.,  $k$  possible tokens with the highest probabilities) for each

predicted token. Formally, we can formulate the uncertainty estimation in our study as

$$u = g(f(\cdot), X, Y, P), \quad (1)$$

where  $P$  is either the probability distribution or top- $k$  probabilities for each token.

Finally, we select twelve uncertainty estimation methods covering *single deterministic methods*, *Bayesian methods*, and *test-time augmentation methods* (see discussion in Sec. II-B). We categorize our uncertainty estimation methods based on the number of inferences required and detail them in the following.

### B. Single-inference Uncertainty Estimation

Single-inference uncertainty estimation methods can be seen as *single deterministic methods* in Sec. II-B. These methods usually calculate an ML model’s confidence based on the probability distribution of the prediction [89], [90]. In particular, such methods are usually used for classification tasks. Though LLMs can be used in various different tasks, the generation of each token can still refer to a classification problem (i.e., choose one token from the entire vocabulary). To aggregate the uncertainty information obtained at the token level, Manakul et al. [29] propose four different metrics to aggregate token-level uncertainty into sentence level.

In particular, a sentence-level uncertainty score can be obtained by taking either the maximum or average of the likelihood  $-\log p$  in a sentence:

$$\text{Max}(-\log p)_i = \max_j(-\log p_{ij}), \quad (2)$$

$$\text{Avg}(-\log p)_i = -\frac{1}{J} \sum_j \log p_{ij}, \quad (3)$$

where  $p_{ij}$  is the probability of a token at position  $j$  of a sentence  $i$ . Additionally, one can also replace the likelihood  $-\log p$  with the entropy  $\mathcal{H}$ :

$$\text{Max}(\mathcal{H})_i = \max_j[\mathcal{H}_{ij}], \quad (4)$$

$$\text{Avg}(\mathcal{H})_i = \frac{1}{J} \sum_j \mathcal{H}_{ij}, \quad (5)$$

where  $\mathcal{H}_{ij}$  is the entropy of this token’s probability distribution over the vocabulary.

After obtaining sentence-level uncertainty estimation, one can further calculate the passage-level uncertainty score by taking the average over all sentence-level uncertainty scores. In this study, we use the metrics discussed above as the **single-inference uncertainty estimation methods** and denote them as **Max Prob**, **Average Prob**, **Max Ent** and **Average Ent**.

### C. Multi-inference Uncertainty Estimation

Multi-inference uncertainty estimation methods leverage the stochastic in either a model’s parameters (e.g., *Bayesian methods*) or data (e.g., *test-time data augmentation methods*) to collect a set of non-deterministic predictions. A model’s prediction uncertainty is then estimated as the divergence among those predictions.

1) *Metrics*: Two metrics were widely used to measure such divergence: (1) variation ratio (VR) and (2) variation ratio for original prediction (VRO) [77], [91], [44]. Originally, both metrics are defined for a classification problem. Wang et al. [92] extend the definitions of VR/VRO and show that they are still effective in tasks other than classification. We introduce these two metrics in the following:

$$VR = 1 - \frac{\sum_{i=1}^T w * \frac{\sum_{j=1, j \neq i}^T (1 - dist(p_i, p_j))}{T-1}}{T}, \quad (6)$$

$$VRO = 1 - \frac{\sum_{i=1}^T (1 - dist(p_i, p_{LM}))}{T}, \quad (7)$$

where  $T$  is the number of inferences,  $dist(\cdot)$  denotes the distance function between two outputs.  $p_i$  and  $p_j$  are the inference result at  $i$ th and  $j$ th inference.  $p_{LM}$  is the prediction result from the original model  $M$ .  $w$  denotes a weight matrix.

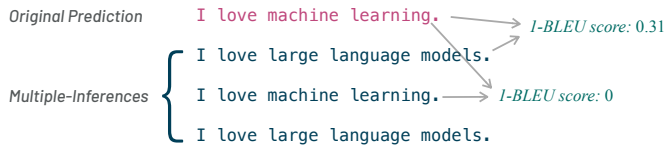


Fig. 5: An example of multiple inferences with LLMs.

We now use an example to demonstrate the calculation of VR/VRO in Fig. 5. Suppose we have three different generations given stochasticity in either model or data.  $(1 - BLEU)$  score [93] is used as  $dist(\cdot)$  to measure the differences between two sentences<sup>1</sup>. Suppose we use identical weights for equation 6, then:

$$VR = 1 - \frac{\frac{0.69+1}{2} + \frac{0.69+0.69}{2} + \frac{0.69+1}{2}}{3} \approx 0.21. \quad (8)$$

Similarly, VRO is

$$VRO = 1 - \frac{0.69 + 0.69 + 1}{3} \approx 0.21. \quad (9)$$

2) *Stochastic inference*: To enable the stochasticity of an ML model, one popular way is to execute it with the dropout layer(s) enabled at test-time [77]. However, this is not feasible given that we focus on “black-box” analysis. We introduce two different ways to enable stochasticity in our study: (1) **sample-based method** (*Bayesian methods*) and (2) **perturbation-based method** (*test-time augmentation methods*).

**Sample-based method.** Randomness exists in LLMs’ generation process. Specifically, by controlling the parameter *temperature* ( $t$ ), an LLM’s generation can be either deterministic or non-deterministic. When  $t$  is 0, an LLM will always choose the token with the largest probability (i.e., greedy decoding). Thus the generation will be deterministic. When  $t > 0$ , an LLM will randomly choose a token as long as its probability is larger than a threshold. A higher  $t$  will lead to a larger randomness in an LLM’s generation. In this study, we set  $t > 0$  to enable an LLM’s stochasticity and generate non-deterministic outputs. We refer to this method as *sample-based method*. We denote

the usages of two different metrics as **Sample VR** and **Sample VRO**. We detail the choices of  $t$  in Sec. IV-D.

**Perturbation-based method.** An LLM’s stochasticity could also be enabled when changing the input prompt or perturbing a generated token. This also refers to the *test-time augmentation methods*. Intuitively, along the chain of the token generation, any perturbation of a generated token can affect the proceeding tokens’ generation, and possibly result in two semantically different texts. We refer to an example in Fig. 3 (C), where LLM answers the question, “How long did the Roman Empire last?” Changing the word *over* in the generation results could yield a completely different result, “lasted for *centuries*.” This phenomenon shows an intriguing property of LLMs: their stochastic nature along the prediction chain with respect to perturbation.

Therefore, in this study, we propose an adaptive method to perturb “critical” token(s) during an LLM’s generation process. Prior work has shown that tokens with high entropy in a sentence usually convey more information from the information theory perspective [94]. Consequently, perturbations at such points could possibly lead to a more pronounced change compared with others. To explore the effectiveness of using such perturbation for measuring uncertainty, we define three types of interest points to perturb: (1) the point with the highest entropy; (2) the point with the lowest entropy; (3) the point that gains the maximum entropy from the previous token in a response. In our study, we select the token that matches one of these three types of points and replace it with  $k$  other tokens with the highest probabilities. We name this method *perturbation-based method* and denote six different variants as: **Max VR**, **Max VRO**, **Min VR**, **Min VRO**, **MaxDiff VR**, and **MaxDiff VRO**. We justify the choice of  $k$  in Sec. IV-D.

## IV. STUDY DESIGN

In this section, we introduce our study design and methodology to answer our research questions in Sec. I.

### A. Subject LLMs

In order to provide a thorough evaluation of the effectiveness of uncertainty measurement as the risk assessment for LLMs in both natural language and domain-specific (i.e., code-generation) problems, we have chosen a broad spectrum of representative models. We selected these models considering their availability, diversity, and computational requirements. Four open-source LLMs for NLP tasks, four closed-source general LLMs, and four LLMs specified for code generation are selected. We present our selected LLMs in Table I and Table II. We additionally performed experiments on older LLMs including GPT-2-XL [60], LLaMA-7B [5], Incoder [33], SantaCoder [34] and CodeGen [35]. Due to space limits, we left related results on our website.

### B. Tasks

To comprehensively understand the effectiveness of selected uncertainty estimation methods, we selected a set of diverse and challenging tasks as a benchmark (Table. III). Specifically,

<sup>1</sup>A higher  $(1 - BLEU)$  indicates a lower similarity between two texts.

TABLE I: Subject open-source LLMs in our study.

LLMs	LLaMA2 [6]	LLaMA3 [95]	Gemma2 [96]	Phi3 [97]	CodeQwen1.5 [98]	DeepSeek Coder [99]	StarCoder 2 [100]	Code Llama [2]
<b>Model Size</b>	6.7B	8B	9B	7B	7B	6.7B	7B	6.7B
<b>Training Data</b>	2T tokens	15T tokens	8T tokens	4.8T tokens	2.4T + 90B	2T	3.5T	2T + 620B
<b>Domain</b>	Text	Text	Text	Text	Code	Code	Code	Code
<b>Provider</b>	MetaAI	MetaAI	Google	Microsoft	Alibaba	DeepSeek	BigCode	MetaAI

TABLE II: Subject closed-source LLMs in our study.

LLMs	GPT-3 [47]	GPT-3.5 [88]	GPT4o [101]	GPT4o mini [102]
<b>Model Size</b>	6.7B	175B	Unknown	Unknown
<b>Training Data</b>	570 GB	Unknown	Unknown	Unknown
<b>Domain</b>	Text	General	Multimodal General	Multimodal General
<b>Provider</b>	OpenAI	OpenAI	OpenAI	OpenAI

TABLE III: The collected NLP and programming tasks.

Dataset	Task Domain	Size	<i>dist</i> (·) metrics in VR/VRO
<b>Eli5-Category</b>	Question Answering	5,411	F1 score [103]
<b>Wiki-QA</b>	Question Answering	243	F1 score [103]
<b>CNN/Daily Mail</b>	Text Summarization	11,490	ROUGE-L [104]
<b>WMT 2014</b>	Machine Translation	3,003	BLEU [93]
<b>MBPP</b>	Code Generation	500	CodeBLEU [105]
<b>HumanEval</b>	Code Generation	164	CodeBLEU [105]

our evaluation covers four NLP tasks from three different domains (i.e., question answering, text summarization, and machine translation). Furthermore, our evaluation also includes two different code-generation tasks. Below, we introduce the tasks of different domains.

**Question Answering (QA)** requires an LLM with the capability of understanding users’ intentions, extracting learned knowledge, and organizing responses. In this study, we selected two different benchmarks for QA: *Eli5-category* [106] and *Wiki-QA* [103]. *Eli5-category* is a collection of 5,411 questions and answers gathered by and obtained from pushshift.io. *Wiki-QA* is a collection of questions extracted from *Bing* query logs with Wikipedia summaries as answers. We selected 243 instances after the de-duplication and removal of questions without answers.

**Text Summarization** aims to condense a long text into a concise short summary. Different from QA, text summarization focuses on benchmarking an LLM’s capabilities of extracting critical information from a long paragraph of text. We use the CNN/Daily Mail dataset [107] as the benchmark for text summarization. It comprises 11,490 news articles from CNN and the Daily Mail, with summaries obtained through the concatenation of highlighted sentences as composed by the original authors. Note that during the evaluation, we add a prompt  $\text{TL;DR}$  after the input text to enable an LLM’s in-context learning ability for summarization [60].

**Machine Translation** is another fundamental task to benchmark language models. In this study, we use the WMT 2014 dataset [108], including 3,003 pairs of French-English transcripts, to evaluate LLMs. Similar to text summarization, We used the prompt template from the GPT-3 paper [47], where a random example translation is presented in the input for in-context learning.

**Code Generation** requires an LLM to understand both natural language (e.g., task description) and programming languages (e.g., formal structures and precise syntax). We chose it as a representative task for evaluating LLMs’ coding abilities because (1) it is a comprehensive task that challenges the models’ advanced understanding and reasoning capabilities, (2) it has been selected as one of the major tasks to benchmark LLMs’ code abilities [2], [98], [99], [100], and (3) it can be automated and objectively evaluated through test cases. We select two datasets as benchmarks for code generation: HumanEval [109] and MBPP [110]. HumanEval consists of 164 programming problems with manually written test cases released by OpenAI. MBPP includes 1,000 crowd-sourced Python programming problems with entry-level difficulties.

### C. Evaluation Metrics

**NLP Tasks.** We use *semantic distance* to measure the performance of LLMs’ generation for NLP tasks. Specifically, we first embed the text using sentence-transformer [111] and compute the cosine distance of the embeddings. A higher cosine distance value indicates a greater level of similarity. This metric can be generalized to different NLP tasks regardless of the length and form of generation. Note that while there are some other metrics specialized for each NLP task, which relies on string-matching (e.g., F1 score for text summarization), we argue two major shortcomings exist when using such metrics. First, string-matching might not accurately capture the divergence between the model’s output and the ground truth (e.g., when the output and ground truth are semantically equivalent while lexically different). Additionally, LLMs without fine-tuning might generate responses in a free-form manner that are often longer than the ground truth in, e.g., text summarization and question answering. In this case, even if an LLM answers correctly, metrics based on string matching could still mis-indicate its performance due to the length differences.

**Code Generation Task.** Different from NLP tasks, it is relatively easy to assess the quality of generated code. We introduce a quality score  $Q$  given by  $Q = (Q_{syntax} + Q_{semantics})/2$ .  $Q_{syntax} = 1$  if the generated code is syntactically correct (i.e., it can be executed by the interpreter). Meanwhile,  $Q_{semantics}$  is the proportion of test cases the generated code passes.

### D. Experiment Settings

**Uncertainty measurement.** We investigate twelve uncertainty estimation methods in our experiments: four *single-inference* methods, two *sample-based* methods, and six *perturbation-based* methods. We set the number of inferences  $T$  to 5 for both *sample-based* and *perturbation-based* methods. This is

for a fair comparison between these two methods since closed-source LLMs (i.e., GPT-3, GPT-3.5) only provide access to top-5 tokens for each token’s generation. Therefore, only 5 inferences can be obtained with the *perturbation-based* method on closed-source LLMs. For temperature  $t$ , we follow the previous work and set  $t$  to 0.7 when enabling on an LLMs’ stochasticity [112], [113].

For distance metric in *VR* and *VRO* (i.e.,  $dist(\cdot)$  in Eq. 6&7), we consider both (1) general metrics based on embeddings’ cosine distance and (2) task-specific metrics. For general metrics, we use *all-mpnet-base-v2* [114] for natural language tasks and *codebert-base* [57] for code data. For task-specific metrics, we consider the following choices in Table III.

**Experiment setups.** Due to the API constraints, an inference of closed-source LLMs (i.e., GPT3 and GPT3.5) could take up to 20 seconds. Consequently, evaluating an LLM on CNN/Daily Mail dataset with 11,490 instances would require nearly 575 hours. Therefore, we randomly sample each NLP task’s dataset with 100 instances when evaluating closed-source LLMs. We further sample 40 out of 164 and 125 out of 500 instances for HumanEval and MBPP, respectively. For closed-source models, we evaluated all instances.

**Hardware and software dependencies.** To conduct our large-scale experiments, we utilize a server with AMD 3955WX CPU (3.9GHz), 256GB RAM, and four NVIDIA A4000 GPUs (16GB VRAM of each). The evaluation of open-source LLMs shown in the main text takes more than 2560 GPU hours, and the results on the website take around 864 GPU hours.

## V. RESULTS

### A. *RQ1: Uncertainty Estimation for NLP Tasks*

To answer this research question, we evaluated eight general LLMs. We present the results of Pearson correlation coefficients between uncertainty scores and LLMs’ performance in Table IV. Such correlation is an indicator of whether the uncertainty estimation can predict LLMs’ performance and further perform the risk assessment. The higher the absolute value of the coefficient, the stronger the correlation. We investigate the results of NLP tasks from three perspectives:

**Uncertainty Measurement Techniques.** As can be observed from Table IV, uncertainties estimated via *sample-based* methods generally yield the highest correlation to an LLM’s performance.

Specifically, *sample-based* methods perform the best, achieving the highest correlation in 21 out of 32 scenarios across eight models and four tasks. Among these, Sample VRO leads in 12 scenarios and Sample VR in 9. In contrast, *single-inference* methods only achieve one top position. This indicates that relying on single-inference uncertainty estimation might be unreliable in practical applications without a more refined strategy.

However, even for the most promising *sample-based* methods, the majority correlation is below -0.6 (18 out of 21 cases) and their performance can still further space for enhancement (e.g., -1.00 correlation), calling for the design of more advanced techniques.

**Finding 1:** *Sample-based VRO* achieves the best performance for eight different LLMs on NLP tasks in most cases, surpassing single-inference methods by a large margin. While its potential is evident, further enhancement with advanced technique design is needed and promising for industrial deployment.

*Perturbation-based* methods demonstrate moderate effectiveness, exceeding single-inference metrics in a substantial proportion of tasks. It reaches the highest correlation in 10 out of 31 scenarios. For the selection of perturbation points, the maximum entropy point is usually better than the other two, counting for 50% of the best cases. Surprisingly, the minimum entropy point can sometimes work well for Llama2. We explored this issue further and found that Llama2 exhibits greater certainty compared to Llama1 and Llama3. For Llama2, we observed that it is substantially more “certain” than Llama-1. The 75th percentile of the entropy of all tokens generated by Llama2 is nearly 0, with a mean value of 0.18. Conversely, Llama-1 has a value of 2.62 at the 75th percentile, with a mean of 1.52. Llama-3 has a value of 0.391 at the 75th percentile, with a mean of 0.211. As such, the entropy selection strategy might not work well on Llama2.

Another interesting observation is that some models (e.g., LLaMA2, LLaMA3, Gemma2, Phi3, and also older models such as GPT-2 and LLaMA demonstrated on our website) exhibit high positive Pearson correlation coefficients of perturbation-based methods. The results indicate that, a higher uncertainty may even yield a better performance in such cases. Upon several case studies, we find that this observation could be attributed to the long input in the text summarization dataset. These LLMs may lose focus on the input context, leading to a lack of understanding of the instructions for summarization. Instead, they mechanically continue the text immediately after the input, producing similar responses despite the stochasticity introduced by perturbations. This results in a low uncertainty score. On the contrary, if an LLM understands the task description at the end of the prompt, it will respond to the perturbation more severely, causing a higher uncertainty.

**Finding 2:** Perturbation-based methods are more inclined to produce model and task-specific outcomes. Stakeholders might need to perform model-specific optimization if possible.

**Influence of Distance Functions.** We also find that task-specific distance does not provide better results compared with cosine distance between embeddings in both *sample-based* and *perturbation-based* methods. On average, using cosine distance achieves an increase of 0.034 for *perturbation-based* methods and an increase of 0.137 for *sample-based* methods across different tasks and LLMs.

**Finding 3:** The cosine distance function yields a better performance on NLP tasks when leveraging stochasticity to estimate an LLM’s prediction uncertainty.



TABLE IV: Pearson correlation coefficients between uncertainty scores and LLMs’ performance on four NLP tasks. The results of VR/VRO are presented as “*cosine distance-based (task-specific distance-based)*.” Highest correlations from different categories are ranked and highlighted as *top-1*, *top-2*, and *top-3*.

Dataset	LLM	Single-inference Method				Sample-based Method	
		Max Prob	Average Prob	Max Ent	Average Ent	Sample VR	Sample VRO
CNN/Daily Mail	LLaMA2	-0.096	-0.109	-0.139	-0.110	-0.370(-0.260)	-0.534(-0.294)
	LLaMA3	0.109	-0.342	-0.002	-0.433	-0.643(-0.510)	-0.590(-0.307)
	Gemma2	-0.331	-0.336	-0.211	-0.317	-0.272(-0.025)	-0.486(-0.204)
	Phi3	-0.535	-0.620	-0.606	-0.648	-0.709(-0.651)	-0.739(-0.648)
	GPT3	-0.231	-0.202	-0.145	-0.170	-0.229(-0.100)	-0.394(-0.244)
	GPT3.5	-0.119	-0.119	-0.036	-0.106	-0.223(-0.131)	-0.220(-0.158)
	GPT4o mini	-0.075	-0.196	-0.021	-0.211	-0.199(-0.187)	-0.177(-0.164)
	GPT4o	-0.334	-0.150	-0.196	-0.126	-0.383(-0.338)	-0.324(-0.290)
	Eli5-Category	LLaMA2	-0.000	-0.008	0.008	-0.018	-0.196(-0.006)
LLaMA3		-0.007	-0.046	-0.043	-0.067	-0.173(-0.051)	-0.177(-0.064)
Gemma2		0.212	0.195	0.168	0.201	0.001(-0.088)	-0.099(-0.106)
Phi3		-0.007	-0.008	-0.044	-0.027	-0.002(0.088)	-0.140(-0.013)
GPT3		-0.054	-0.207	-0.016	-0.172	-0.236(-0.084)	-0.382(-0.140)
GPT3.5		-0.037	-0.240	-0.084	-0.289	-0.168(-0.021)	-0.223(-0.046)
GPT4o mini		-0.200	-0.283	-0.099	-0.303	-0.116(-0.162)	-0.127(-0.064)
GPT4o		0.019	-0.179	0.007	-0.207	-0.224(-0.146)	-0.152(-0.118)
Wiki-QA		LLaMA2	-0.181	-0.193	-0.166	-0.140	-0.281(-0.206)
	LLaMA3	0.017	0.091	-0.011	0.093	-0.227(-0.060)	-0.294(-0.050)
	Gemma2	0.226	0.265	0.259	0.300	-0.031(0.213)	-0.138(0.212)
	Phi3	0.038	-0.101	-0.004	-0.101	-0.395(-0.045)	-0.370(0.030)
	GPT3	-0.112	-0.192	0.017	-0.107	-0.326(-0.061)	-0.376(-0.139)
	GPT3.5	-0.051	-0.107	-0.065	-0.105	-0.401(-0.085)	-0.425(-0.079)
	GPT4o mini	0.032	0.031	-0.018	0.011	-0.223(-0.051)	-0.124(0.010)
	GPT4o	0.080	-0.052	0.006	-0.083	-0.222(-0.074)	-0.128(-0.054)
	WMT 2014	LLaMA2	0.157	-0.000	0.185	-0.015	0.068(0.009)
LLaMA3		-0.113	-0.001	-0.139	-0.020	0.006(0.105)	-0.071(-0.001)
Gemma2		0.342	0.153	0.309	0.115	-0.510(-0.410)	-0.769(-0.485)
Phi3		-0.024	-0.126	-0.071	-0.09	-0.393(-0.159)	-0.392(-0.164)
GPT3		-0.046	-0.162	-0.039	-0.176	-0.250(-0.254)	-0.251(-0.339)
GPT3.5		0.089	-0.158	0.050	-0.175	-0.244(-0.065)	-0.233(-0.017)
GPT4o mini		-0.223	-0.136	-0.138	-0.122	-0.403(-0.056)	-0.394(-0.095)
GPT4o		-0.054	-0.273	-0.06	-0.265	-0.558(-0.209)	-0.539(-0.137)
Dataset		LLM	Perturbation-based				
	Max VR		Max VRO	Min VR	Min VRO	MaxDiff VR	MaxDiff VRO
CNN/Daily Mail	LLaMA2	-0.204(-0.161)	-0.388(-0.154)	-0.075(-0.055)	-0.418(-0.226)	-0.194(-0.152)	-0.378(-0.143)
	LLaMA3	0.027(0.005)	0.123(0.231)	0.472(0.385)	0.181(0.204)	0.076(0.043)	0.167(0.256)
	Gemma2	0.405(0.488)	0.308(0.435)	0.296(0.304)	0.378(0.422)	0.443(0.474)	0.404(0.461)
	Phi3	0.447(0.555)	0.579(0.669)	0.343(0.446)	0.511(0.598)	0.437(0.551)	0.570(0.659)
	GPT3	-0.096(-0.084)	-0.233(-0.209)	-0.001(-0.049)	0.034(0.041)	-0.119(-0.121)	-0.228(-0.182)
	GPT3.5	0.064(-0.086)	-0.024(-0.087)	-0.116(-0.176)	-0.068(-0.177)	-0.042(-0.138)	0.072(0.021)
	GPT4o mini	-0.227(-0.288)	0.065(0.058)	0.085(0.020)	0.058(0.048)	-0.118(-0.240)	0.138(0.111)
	GPT4o	-0.110(-0.119)	-0.086(-0.048)	0.064(0.041)	0.060(0.049)	-0.092(-0.098)	-0.043(0.011)
	Eli5-Category	LLaMA2	-0.079(0.008)	-0.203(-0.017)	-0.086(0.002)	-0.218(0.026)	-0.086(0.018)
LLaMA3		-0.106(-0.056)	-0.159(-0.032)	-0.062(-0.070)	-0.104(-0.091)	-0.093(-0.055)	-0.159(-0.032)
Gemma2		-0.381(-0.172)	-0.462(-0.206)	-0.089(-0.004)	-0.158(-0.074)	-0.398(-0.179)	-0.519(-0.212)
Phi3		-0.208(-0.108)	-0.231(-0.058)	-0.120(-0.070)	-0.169(-0.073)	-0.235(-0.122)	-0.231(-0.102)
GPT3		0.168(0.179)	0.026(0.079)	-0.236(-0.246)	-0.066(-0.135)	0.140(-0.003)	0.184(0.047)
GPT3.5		0.003(-0.076)	0.065(0.049)	-0.087(-0.157)	0.072(0.056)	0.160(0.092)	0.183(0.125)
GPT4o mini		-0.109(-0.045)	-0.339(-0.053)	-0.054(0.002)	-0.222(-0.150)	-0.131(-0.008)	-0.159(0.047)
GPT4o		0.115(0.108)	-0.017(0.144)	-0.158(-0.020)	-0.073(0.080)	-0.026(-0.036)	-0.001(-0.047)
Wiki-QA		LLaMA2	-0.057(0.050)	-0.081(0.014)	-0.064(-0.001)	-0.164(-0.036)	-0.077(0.023)
	LLaMA3	-0.043(0.022)	-0.037(0.071)	-0.118(-0.114)	-0.154(-0.147)	-0.068(-0.014)	-0.032(0.081)
	Gemma2	-0.255(0.103)	-0.480(-0.034)	-0.094(0.301)	-0.233(0.163)	-0.561(-0.123)	-0.634(-0.118)
	Phi3	-0.067(-0.001)	-0.132(0.002)	-0.152(-0.038)	-0.238(-0.033)	-0.181(-0.118)	-0.179(-0.083)
	GPT3	-0.064(0.049)	-0.204(-0.186)	-0.042(0.048)	-0.247(-0.146)	-0.042(-0.020)	-0.095(-0.111)
	GPT3.5	0.092(0.065)	0.078(0.056)	-0.017(-0.113)	-0.030(-0.109)	0.157(0.014)	0.214(0.065)
	GPT4o mini	-0.263(-0.196)	-0.184(-0.116)	-0.004(0.053)	0.014(0.008)	-0.150(-0.144)	-0.103(-0.110)
	GPT4o	0.023(-0.048)	0.135(0.075)	-0.000(0.027)	-0.071(-0.047)	-0.060(-0.076)	0.081(0.055)
	WMT 2014	LLaMA2	-0.022(-0.027)	-0.036(0.058)	-0.064(-0.029)	-0.190(0.024)	-0.026(-0.028)
LLaMA3		0.243(0.369)	0.153(0.257)	-0.111(-0.090)	-0.335(-0.346)	0.281(0.429)	0.293(0.362)
Gemma2		0.027(0.049)	0.243(0.100)	-0.016(0.048)	0.326(0.266)	0.052(0.027)	0.183(0.068)
Phi3		-0.109(-0.102)	-0.060(-0.039)	-0.064(-0.088)	-0.033(-0.040)	-0.125(-0.123)	-0.052(-0.040)
GPT3		-0.076(0.006)	0.016(0.021)	-0.033(0.095)	0.215(0.138)	-0.103(0.001)	0.013(0.040)
GPT3.5		-0.277(-0.363)	-0.101(-0.081)	-0.099(-0.048)	-0.007(-0.003)	-0.105(-0.201)	-0.031(-0.024)
GPT4o mini		-0.001(-0.010)	-0.116(-0.030)	-0.091(-0.058)	-0.133(-0.081)	-0.043(-0.034)	-0.064(-0.015)
GPT4o		-0.003(0.018)	0.116(0.161)	-0.058(-0.087)	-0.062(-0.069)	0.060(-0.051)	0.102(0.080)

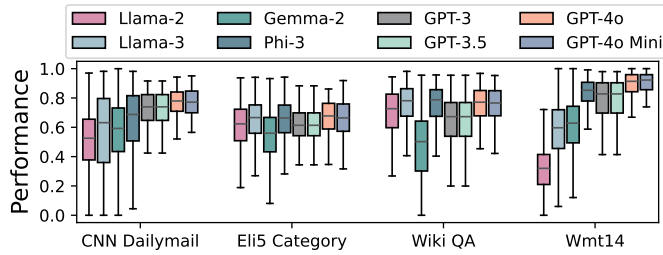


Fig. 6: Performance of LLMs on NLP tasks

**Influence of Models.** Combining LLMs’ performance in Figure 6 and uncertainty methods’ effectiveness in Table IV, we can observe that it is challenging for these uncertainty methods to assess the risks for models that are either *exceptionally good* (e.g., GPT4o) or *notably poor* (e.g., Llama2). This might be because the current evaluated methods struggle to discern the nuanced differences attributed to aleatoric uncertainty [27] (knowledge deficiency). Most of the results are likely dominated by epistemic uncertainty [27] (inherent randomness), leading to a decline in detection performance. In other words, when generated contents are unanimously good or bad, the computed uncertainty is more likely driven by inherent randomness rather than variations in the input data.

Another intriguing observation is that single-inference methods tend to perform better on more advanced models, such as the closed-source GPT series. This may be due to their extensive training data, resulting in better-calibrated confidence scores. However, a gap remains between single-inference methods and other approaches. Bridging this gap is challenging, as LLM output entropy consists of two components [115]: language entropy, which arises from multiple vocabulary choices that convey similar meanings, and excess cross-entropy, which reflects model capability. Since single-inference methods rely on output entropy, separating these components is difficult. In contrast, multi-inference methods mitigate this issue by focusing on semantic variability rather than output entropy alone.

**Finding 4:** Uncertainty-based risk assessments for exceptionally good or notably poor LLMs appear to be limited. The former is especially prevalent in closed-source commercial models used in real-world applications. When an LLM’s performance is very high, analyzing its potential issues can be intrinsically difficult, calling for more advanced methods that can work even with limited information.

### B. RQ2: Limitations in Uncertainty Estimation for NLP Tasks

In this RQ, we further explore two limitations of evaluated uncertainty estimation methods.

**Influence of prompt.** An intriguing anomaly in RQ1 is Llama2, which underperforms significantly on summarization and translation, and its uncertainty estimation for all four tasks is ineffective. This is counter-intuitive. After manually investigating some randomly chosen samples, we found this might be because of the input prompt. Using a fine-tuned chat

version of Llama2, we incorporated a system prompt template shown in the original paper, starting with

```
You are a helpful, respectful, and
honest assistant ...
```

However, this prompt template has three downsides. We will use the WMT-14 translation task as examples: (1) The LLM will sometimes directly reject to respond, offering something such as “I cannot provide a translation for that statement as it is not factually coherent.” (2) The LLM misinterprets examples in the in-context prompt, translating the (fixed) example itself. This leads to duplicate translations and results in poor translation performance. (3) Many of the responses will begin with greetings such as “Thank you for your kind and respectful instructions!” Each of these three scenarios can negatively affect the calculation of multi-inference uncertainty, as the fine-tuned responses from the LLM, to some extent, surpass inherent randomness.

To study the prompt influence further, we removed the system prompt and re-run all the experiments on Llama2. The results are shown in Table VI. Overall, there’s a significant increase in all correlations, encompassing the original two high-performing QA tasks. When comparing the top-1 scores across the four tasks, the average increase in the absolute value of correlation coefficients is 0.272. The most notable improvement is observed in the WMT-14 dataset, where the correlation shifts from -0.190 to -0.746. Given that RLHF-based prompt tuning is pivotal for SOTA chat models, such as open-sourced Llama2 and commercial ChatGPT, it might be better for future work to take this into consideration when designing new methods.

**Finding 5:** Prompts could significantly influence uncertainty estimation in some cases. Prompt templates can lead LLMs to exhibit different behaviors, compromising the accuracy of uncertainty estimation.

**Relation of uncertainty and inaccuracy.** We find that a low uncertainty does not guarantee that an LLM’s response is reliable. Specifically, an LLM can generate highly confident responses while with non-factual information.

A representative example can be observed when Gemma2 is instructed to summarize a CNN news article in which a triathlon participant narrates her training journey in the first person.

```
... I’m looking forward to many more
rides outdoors. I want to say thank
you to the fellowship of fitness that
I’ve been lucky enough to find ... I’m
so grateful that ...
```

The article contains many first-person narratives, which cause the model to misinterpret the task and ignore the instruction to summarize. As a result, it repeated sentences such as “I’m still not doing as much as xxx as I should ...” In the response, the LLM consistently produced similar responses, even when perturbation or sampling was introduced to add stochasticity. Despite the fact that these responses were

TABLE V: AUC scores for detecting erroneous code generated by LLMs. The results of VR/VRO are presented as “*cosine distance-based (task-specific distance-based)*.” Highest scores are ranked and highlighted as **top-1**, **top-2**, and **top-3**.

Dataset	LLM	Single-inference Method				Sample-based Method	
		Max Prob	Average Prob	Max Ent	Average Ent	Sample VR	Sample VRO
HumanEval	CodeQwen1.5	0.592	0.539	0.578	0.538	0.639(0.617)	<b>0.668(0.647)</b>
	DeepSeek Coder	<b>0.735</b>	0.692	<b>0.738</b>	0.676	0.730(0.705)	<b>0.747(0.699)</b>
	StarCoder2	<b>0.747</b>	0.629	<b>0.752</b>	0.656	0.693(0.675)	<b>0.748(0.670)</b>
	Code Llama	<b>0.744</b>	0.618	<b>0.752</b>	0.601	0.697(0.681)	<b>0.731(0.749)</b>
	GPT3.5	0.500	0.590	0.383	0.490	0.630( <b>0.667</b> )	<b>0.607(0.587)</b>
	GPT4o mini	<b>0.909</b>	<b>0.760</b>	<b>0.891</b>	0.749	0.657(0.583)	0.737(0.714)
	GPT4o	0.583	0.623	0.480	0.543	0.423(0.623)	0.343(0.606)
MBPP	CodeQwen1.5	0.356	0.363	0.358	0.367	0.555( <b>0.613</b> )	<b>0.593(0.643)</b>
	DeepSeek Coder	0.545	0.463	0.527	0.46	0.581(0.618)	0.617(0.621)
	StarCoder2	0.567	0.447	0.589	0.439	0.562(0.616)	0.581( <b>0.665</b> )
	Code Llama	0.499	0.418	0.483	0.433	0.578( <b>0.668</b> )	0.603( <b>0.705</b> )
	GPT3.5	<b>0.582</b>	0.556	<b>0.606</b>	0.560	0.592(0.563)	0.561(0.576)
	GPT4o mini	0.579	0.609	0.593	<b>0.628</b>	0.609(0.606)	0.599(0.569)
	GPT4o	0.629	0.633	0.635	<b>0.667</b>	0.650( <b>0.735</b> )	0.634( <b>0.698</b> )

Dataset	LLM	Perturbation-based					
		Max VR	Max VRO	Min VR	Min VRO	MaxDiff VR	MaxDiff VRO
HumanEval	CodeQwen1.5	0.629(0.611)	<b>0.646(0.609)</b>	0.595(0.592)	0.593(0.571)	0.626(0.627)	0.638(0.595)
	DeepSeek Coder	0.607(0.479)	0.664(0.512)	0.589(0.563)	0.574(0.426)	0.604(0.454)	0.661(0.581)
	StarCoder2	0.638(0.569)	0.664(0.643)	0.449(0.457)	0.565(0.526)	0.592(0.525)	0.666(0.629)
	Code Llama	0.597(0.613)	0.668(0.644)	0.498(0.555)	0.643(0.582)	0.609(0.538)	0.725(0.680)
	GPT3.5	0.500(0.503)	0.457(0.560)	0.497(0.503)	0.420( <b>0.607</b> )	0.533(0.533)	0.393(0.477)
	GPT4o mini	0.546(0.531)	0.703(0.743)	0.594(0.531)	0.589(0.646)	0.506(0.460)	0.686(0.749)
	GPT4o	0.571(0.560)	<b>0.709(0.720)</b>	0.391(0.469)	0.600(0.674)	0.457(0.509)	<b>0.691(0.680)</b>
MBPP	CodeQwen1.5	0.533(0.532)	0.501(0.553)	0.476(0.478)	0.487(0.520)	0.567(0.553)	0.523(0.558)
	DeepSeek Coder	0.620(0.592)	0.561(0.566)	0.456(0.480)	0.459(0.487)	<b>0.658(0.654)</b>	0.619(0.619)
	StarCoder2	0.600(0.576)	0.568(0.587)	0.485(0.451)	0.420(0.477)	<b>0.635(0.632)</b>	0.570(0.627)
	Code Llama	0.586(0.630)	0.521(0.621)	0.454(0.507)	0.501(0.568)	0.622( <b>0.679</b> )	0.566(0.667)
	GPT3.5	0.587( <b>0.628</b> )	0.511(0.544)	0.550(0.495)	0.508(0.511)	0.498(0.571)	0.486(0.528)
	GPT4o mini	0.574(0.487)	0.574(0.463)	0.416(0.397)	0.420(0.435)	<b>0.650(0.551)</b>	<b>0.675(0.533)</b>
	GPT4o	0.448(0.538)	0.418(0.546)	0.560(0.591)	0.536(0.549)	0.514(0.545)	0.454(0.495)

TABLE VI: Pearson correlation coefficients of LLaMA2 without system prompts on four NLP tasks. The results of VR/VRO are presented as “*cosine distance-based (task-specific distance-based)*.” Highest correlations from different categories are ranked and highlighted as **top-1**, **top-2**, and **top-3**.

Dataset	Single-inference Method				Sample-based Method	
	Max Prob	Average Prob	Max Ent	Average Ent	Sample VR	Sample VRO
CNN/Daily Mail	-0.349	-0.480	-0.435	-0.528	<b>-0.601(-0.528)</b>	<b>-0.638(-0.522)</b>
Eli5-Category	-0.054	-0.316	-0.182	-0.362	-0.354(-0.020)	-0.444(0.006)
Wiki-QA	-0.093	-0.122	-0.073	-0.173	<b>-0.534(-0.212)</b>	<b>-0.589(-0.138)</b>
WMT 2014	-0.006	0.176	-0.145	-0.003	<b>-0.611(-0.527)</b>	<b>-0.746(-0.578)</b>

Dataset	Perturbation-based					
	Max VR	Max VRO	Min VR	Min VRO	MaxDiff VR	MaxDiff VRO
CNN/Daily Mail	0.098(-0.054)	-0.006(-0.053)	0.000(-0.087)	-0.332(-0.320)	0.123(-0.037)	0.025(-0.028)
Eli5-Category	-0.191(-0.124)	<b>-0.513(-0.293)</b>	-0.183(-0.042)	-0.256(0.059)	-0.169(-0.109)	<b>-0.498(-0.276)</b>
Wiki-QA	0.007(0.029)	-0.008(0.070)	-0.193(-0.123)	<b>-0.438(-0.103)</b>	0.049(0.010)	-0.058(0.011)
WMT 2014	0.034(-0.066)	0.230(-0.032)	-0.400(-0.409)	<b>-0.608(-0.515)</b>	0.030(-0.055)	0.266(0.014)

entirely incorrect, the LLM still generated a low uncertainty score. We provide the full example on our website.

Conversely, a higher degree of uncertainty also does not necessarily imply that an LLM’s prediction is incorrect. We show an example of GPT 3.5, which summarizes a piece of news from the CNN/Daily Mail dataset. In this case, the ground truth summary includes two details: *a fire occurrence at a park and the absence of injuries*. For sample-based uncertainty measurement, all five generated samples incorporated these two pieces of information but also furnished additional varied information, such as the park’s owner and its intended use.

Such extra information further leads to a higher variance in the generated samples’ embedding, resulting in a high uncertainty score despite that the LLM’s prediction is reliable.

**Finding 6:** Uncertainty is not always correlated to inaccuracy. Future work may also consider combining other features or indicators (besides uncertainty) to the risk assessment for better performance.

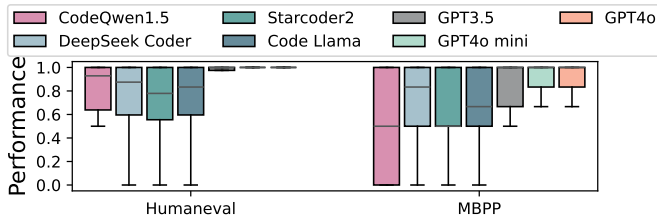


Fig. 7:  $Q$  scores of LLMs on code generation tasks

### C. RQ3: Uncertainty Estimation for Code Generation

To answer this research question, we evaluate seven LLMs capable of code generation. Their performances are shown in Figure 7. When evaluating the efficacy of uncertainty methods in identifying erroneous code, we treat the problem as a binary detection task. Code is labeled as 1 if completely correct (e.g.,  $Q = 1$ ) and 0 if not. We take the uncertainty scores as indicators, and higher uncertainty suggests a greater likelihood of errors in the code. Subsequently, we calculate the Area Under the Receiver Operating Characteristic Curve (AUC) scores for each method. The results are shown in Table V.

**Uncertainty Measurement Techniques.** Even with the distinct task of code generation and the different evaluation metrics of AUC, we can still observe a similar trend on code generation tasks than on NLP tasks, i.e., *sample-based* methods dominate across two different datasets and diverse LLMs. Perturbation-based methods remain in second place, with single-inference methods ranking last overall. However, a nuanced distinction is that single-inference methods perform better in code generation tasks compared to other NLP tasks. Specifically, they achieved 3 top-1 AUC scores out of 14 cases (21.4%), while for NLP tasks, they only secured one top-1 correlation out of 32 cases (3.13%). One possible explanation is that programming languages are more deterministic than natural languages, which reduces the impact of language entropy [115] on uncertainty estimation.

**Finding 7:** Sample-based methods remain the most effective for risk assessment in code generation. Although single-inference methods perform worse than perturbation-based methods, they show stronger performance on code generation tasks compared to NLP tasks.

**Influence of Distance Functions.** Different from the results on NLP tasks, the cosine distance between code embeddings under-performs significantly in comparison with the task-specific distance function, CodeBLEU. Task-specific metric prevails in 14 out of 24 cases for the TOP-3 results.

**Finding 8:** The distance function can play an important role in uncertainty estimation as indicated by the result difference between domains of NLP and code. A more carefully designed distance function that suits downstream tasks could potentially enhance the effectiveness of risk assessment.

**Influence of Models.** Among all models, GPT-3.5 is the most challenging model to assess code quality using uncertainty

estimation. The best AUC score is 0.667 on the HumanEval dataset and 0.628 on the MBPP dataset. Both of them are below 0.7. According to Figure 7, GPT-3.5 demonstrates relatively strong performance compared to most models but still falls short of GPT4o and GPT4o mini. It is expected that GPT-3.5 has a lower AUC score than other open-source models because, as seen in our findings on NLP tasks, nuanced errors are more difficult to detect using uncertainty metrics. The errors made by GPT-3.5 tend to be subtler than those made by open-source models. However, it may seem counterintuitive that assessing risks for GPT-3.5 is more challenging compared to GPT4o and GPT4o mini. Upon further exploration, we found that GPT-3.5 is more prone to making *partial errors*. For instance, on the MBPP dataset, 40.4% of GPT-3.5's errors passed at least one test case, compared to 27.3% for GPT4o and 32.4% for GPT4o mini. This suggests that the GPT4 family is more likely to either fully misunderstand the question or generate a completely correct response, making their errors easier to detect.

### D. RQ4: Limitations in Uncertainty Estimation for Code Generation

**Limitations of detecting subtle errors.** In RQ3, we hypothesized that the relatively poor performance of uncertainty estimation in the detection of GPT-3.5's errors is due to the challenges in pinpointing subtle mistakes that existed in the generated code. To validate this hypothesis, we conducted additional experiments in this RQ. We evaluated all the methods across seven LLMs (with results for three available on our website) on two datasets under two settings. In setting *A*, only syntactically correct code is kept (e.g.,  $Q \geq 0.5$ ). In setting *B*, only syntactically incorrect and completely correct code is kept (e.g.,  $Q = 1$  or  $Q < 0.5$ ). We excluded the GPT family since all their code is syntactically correct. We recorded the best AUC score for each model under both settings. We leave the full result on our website.

Comparing the performances between setting *A* and setting *B*, we observe that, with the exception of Santacoder on HumanEval, in which we exclude details in the main content due to page limit, all other LLMs show a notable improvement. The average score increase is 0.095. Santacoder's decline might be attributed to its frequent generation of syntactically incorrect code, making the partially correct code an easily detectable outlier.

**Finding 9:** Current uncertainty methods are more suitable for detecting obvious errors instead of subtle errors.

**Limitations of distance functions.** Different from our observations in RQ1, the cosine distance metric does not show dominant performance compared with task-specific metrics on estimating LLMs' risks when generating code. This indicates that it is non-trivial for selected embedding model [57] to detect minor code differences due to an LLM's stochasticity.

The efficiency of multi-inference metrics heavily depends on the precise estimation of the distance between data points.

TABLE VII: The average distances between generated code.

	Perturbation		Sample	
	Open-source	Closed-source	Open-source	Closed-source
Cosine	0.030±0.034	0.015±0.036	0.027±0.017	0.009±0.010
CodeBLEU	0.513±0.198	0.260±0.213	0.679±0.111	0.374±0.210

Both RQ1 and RQ3 highlight the crucial role of the distance function. However, it appears that the cosine distance metric falls short in code-related tasks. It is non-trivial for current embedding LLMs to detect subtle differences generated by randomness or perturbations.

We calculated pairwise distances within each group of responses generated via sample-based and perturbation-based methods. As shown in Table VII, the average cosine distance is significantly lower than its task-specific counterpart (i.e., CodeBLEU [105]). This directly affects the effectiveness of *VR/VRO* when determining the degree of uncertainty.

Nevertheless, CodeBLEU [105] also comes with its limitations. For instance, two programs that only differ from variable names might lead to a large CodeBLEU distance, resulting in erroneous uncertainty estimation.

**Finding 10:** Neither cosine distance nor CodeBLEU could accurately assess the difference between the two programs, resulting in under-performed uncertainty estimation for LLMs compared with NLP tasks. Characterizing the true difference between code can be challenging and is a limitation to performing accurate uncertainty estimation.

## VI. IMPLICATION AND OPPORTUNITY

### A. For Researchers

**Prompt is important.** Prompt has long been proven to be a key factor in LLM’s performance. In our study, we further demonstrate its potential to significantly affect uncertainty estimation’s efficacy (Finding 5). Specifically, the uncertain behavior of LLMs might be profoundly impacted by the prompt used in RLHF [30]. Original uncertain answers may be supplanted by human-favored responses when integrating specific prompt templates. A prospective research direction is to explore the influence of the RLHF process on uncertainty estimation and to discern strategies for more accurate estimations, both from training (e.g., calibrate the model better) and inference (e.g., refine estimation methods) standpoints. Another interesting perspective is to design a better prompt (e.g., instruct LLMs to switch to the uncertainty estimation mode) to enhance the precision of the measurement.

**Subtle errors can be hard to detect.** We observed that selected methods could struggle to detect subtle errors in partially correct code and are easier to obvious errors (Finding 9). One possible future direction is to improve their sensitivity by separating the uncertainty caused by model inability and that stemming from inherent randomness. Another direction could

involve constructing a multi-stage system, with uncertainty-related methods at the forefront, followed by other techniques (e.g., white-box).

**Better perturbation strategy is needed for more accurate uncertainty estimation.** Our *perturbation-based* methods leverage the unique characteristic of an autoregressive language model to perturb its decoding process, which shows moderate uncertainty estimation performance in the experiments. Compared with *sample-based* methods, the *perturbation-based* methods do not require access and tuning the temperature setting ( $T$ ). Despite the fact that the *perturbation-based* methods underperform the *sample-based* methods in general, we believe the *perturbation-based* methods could be further improved with, e.g., a more fine-grained strategy to identify key point for perturbation.

**Using uncertainty estimation alone might not be enough for comprehensive risk assessment.** In RQ2 and Finding 6, we discussed and concluded that a model’s uncertainty does not necessarily indicate its correctness. LLMs can be highly confident in incorrect answers and vice versa. Therefore, relying solely on uncertainty measurement may not be sufficient for practical LLM deployment. A promising direction is to incorporate behavioral testing [116], [117], [118], a method recently proposed for evaluating the correctness of NLP systems. Rather than assessing whether LLMs are fully accurate, behavioral testing evaluates whether their output meets certain important properties. This more flexible approach could potentially be automated without the need for ground truth and might help filter out clearly unreasonable responses, such as those that repeatedly generate meaningless sentences. We believe that integrating behavioral testing with uncertainty measurement into a comprehensive risk assessment framework will open new opportunities for improving the reliability of LLM-driven systems.

### B. For Developers

**Ask more, get more.** In our study, *multi-inference* methods perform better than single-inference methods across different tasks in most cases. As for a *black-box* LLM, getting a comprehensive understanding beforehand or only through a single deterministic inference could be challenging. Instead, the more we query an LLM, the clearer we can get about its internal knowledge regarding a specific aspect. We hypothesize that this is because by querying models multiple times, we gain more knowledge from them. This might be a promising technique when the model is black-box.

**Model-specific uncertainty estimation might be beneficial.** Although all the methods chosen in the paper are black-box, we observed considerable variations in their effectiveness across different models (e.g., Finding 2). Thus, to enhance the efficacy of an uncertainty-based risk assessment system, stakeholders might need to tailor their methods and undertake model-specific optimizations. These adjustments could be necessary even between different model versions (e.g., LLaMA2 and LLaMA3). Another important factor that could influence the effectiveness of uncertainty measurement is the deployment choices, such as quantization types and even the



backend hardware. These factors may affect computational accuracy and create subtle variations between the original models. In practical scenarios, models can be versioned based on their quantization level, which might significantly impact their performance and computational efficiency [119]. We believe this is an intriguing direction for future exploration and will leave it for future work.

## VII. THREATS TO VALIDITY

In terms of **internal threats**, the selection of uncertainty estimation techniques can be a threat that affects the validity of our findings. In our study, we tried our best and collect as many as 12 existing uncertainty estimation methods from different categories (single deterministic, Bayesian, and test-time augmentation), to better understand the effectiveness of uncertainty estimation under the scope of LLMs' erroneous generations.

In terms of **external threats**, the subject LLM selection for the evaluation could be another threat. The generalizability of our uncertainty estimation methods to LLMs beyond those studied presents a potential threat, as LLMs may exhibit different uncertainty characteristics across various tasks. Additionally, our findings and conclusions may differ as LLM technology advances. To address this, we selected a diverse range of 17 LLMs from both open-source and closed-source candidates, spanning from GPT-2 XL (released in 2019) to GPT4o (released very recently). We have carefully reviewed our conclusions and made every effort to present findings that are relevant and useful for state-of-the-art LLMs. Still, future models may exhibit different characteristics. To support continued research, we will open-source our codebase, allowing future researchers to test and analyze emerging models as they are developed.

One limitation of this work is that we focus solely on evaluating code generation for code-specific LLMs. Other tasks, such as program repair [120] and code translation [121], also merit consideration. We leave a more in-depth analysis of these code-related tasks for future research.

## VIII. CONCLUSION

This paper initiates an early exploratory study toward understanding the risk assessment of LLMs from the lens of uncertainty estimation. A large-scale evaluation of twelve different uncertainty estimation techniques on nine LLMs and six tasks is conducted. A further in-depth analysis was made to investigate the correlations between LLMs' prediction uncertainty and their performance. Understanding the potential risks of LLMs could be of great importance for industrial-scale applications. Our results confirm that uncertainty estimation can be a promising direction for potential risk assessment of LLMs in both NLP and code-generation domain tasks. However, there can still be much space and opportunity to design more advanced uncertainty estimation techniques to characterize the risks of an LLM more effectively. Moreover, other possibly useful quality indicators besides uncertainty could also be designed to better characterize the capability boundary of an LLM from multiple perspectives. With the

recently increasing demand and urgency for trustworthiness assurance of LLMs in industry, we hope this paper could potentially inspire researchers and practitioners, to join the force to design novel techniques and toolchain support and together conquer many new relevant challenges. We also make the replication package of this paper available, to enable further research towards realizing trustworthy LLMs for industrial usage.

## ACKNOWLEDGMENTS

This work was supported in part by JST CRONOS Grant (No.JPMJCS24K8), JST-Mirai Program Grant (No.JPMJMI20B8), JSPS KAKENHI Grant (No.JP21H04877, No.JP23H03372, and No.JP24K02920), Canada CIFAR AI Chairs Program, the Natural Sciences and Engineering Research Council of Canada, and the Autoware Foundation.

## REFERENCES

- [1] T. Markov, C. Zhang, S. Agarwal, F. E. Nekoul, T. Lee, S. Adler, A. Jiang, and L. Weng, "A holistic approach to undesired content detection in the real world," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 12, 2023, pp. 15 009–15 018.
- [2] B. Rozière, J. Gehring, F. Gloeckle, S. Sootla, I. Gat, X. E. Tan, Y. Adi, J. Liu, T. Remez, J. Rapin *et al.*, "Code llama: Open foundation models for code," *arXiv preprint arXiv:2308.12950*, 2023.
- [3] "Chatgpt," <http://chat.openai.com>, 2023.
- [4] L. Wu, Z. Zheng, Z. Qiu, H. Wang, H. Gu, T. Shen, C. Qin, C. Zhu, H. Zhu, Q. Liu *et al.*, "A survey on large language models for recommendation," *arXiv preprint arXiv:2305.19860*, 2023.
- [5] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.
- [6] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.
- [7] Meta, "The llama ecosystem: Past, present, and future," 2023. [Online]. Available: <https://perma.cc/YJ4E-FB95>
- [8] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung, "Survey of hallucination in natural language generation," *ACM Computing Surveys*, vol. 55, no. 12, pp. 1–38, 2023.
- [9] A. Tamkin, M. Brundage, J. Clark, and D. Ganguli, "Understanding the capabilities, limitations, and societal impact of large language models," *arXiv preprint arXiv:2102.02503*, 2021.
- [10] A. Abid, M. Farooqi, and J. Zou, "Persistent anti-muslim bias in large language models," in *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, 2021, pp. 298–306.
- [11] Y. Bang, S. Cahyawijaya, N. Lee, W. Dai, D. Su, B. Wilie, H. Lovenia, Z. Ji, T. Yu, W. Chung *et al.*, "A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity," *arXiv preprint arXiv:2302.04023*, 2023.
- [12] D. Johnson, R. Goodman, J. Patrinely, C. Stone, E. Zimmerman, R. Donald, S. Chang, S. Berkowitz, A. Finn, E. Jahangir *et al.*, "Assessing the accuracy and reliability of ai-generated medical responses: an evaluation of the chat-gpt model," 2023.
- [13] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg *et al.*, "Sparks of artificial general intelligence: Early experiments with gpt-4," *arXiv preprint arXiv:2303.12712*, 2023.
- [14] J. Schuett, N. Dreksler, M. Anderljung, D. McCaffary, L. Heim, E. Bluemke, and B. Garfinkel, "Towards best practices in agi safety and governance: A survey of expert opinion," *arXiv preprint arXiv:2305.07153*, 2023.
- [15] M. Anderljung, J. Barnhart, J. Leung, A. Korinek, C. O'Keefe, J. Whittestone, S. Avin, M. Brundage, J. Bullock, D. Cass-Beggs *et al.*, "Frontier ai regulation: Managing emerging risks to public safety," *arXiv preprint arXiv:2307.03718*, 2023.
- [16] Microsoft, "Responsible ai standard, v2," 2022. [Online]. Available: <https://perma.cc/S466-T3HD>

- [17] OpenAI, "Our approach to ai safety," 2023. [Online]. Available: <https://perma.cc/4M6Z-LGX6>
- [18] Amazon, "Transform responsible ai from theory into practice," 2023. [Online]. Available: <https://perma.cc/V6LF-ZPD7>
- [19] Google, "Responsible ai practices," 2023. [Online]. Available: <https://perma.cc/SAG5-TEHL>
- [20] L. Koessler and J. Schuett, "Risk assessment at agi companies: A review of popular risk assessment techniques from other safety-critical industries," *arXiv preprint arXiv:2307.08823*, 2023.
- [21] Meta, "Facebook's five pillars of responsible ai," 2021. [Online]. Available: <https://ai.meta.com/blog/facebook-five-pillars-of-responsible-ai/>
- [22] —, "Responsible ai," 2023. [Online]. Available: <https://perma.cc/9BAY-NM9D>
- [23] —, "Building generative ai features responsibly," 2023. [Online]. Available: <https://about.fb.com/news/2023/09/building-generative-ai-features-responsibly/>
- [24] N. Kokhlikyan, V. Miglani, M. Martin, E. Wang, B. Alsallakh, J. Reynolds, A. Melnikov, N. Kliushkina, C. Araya, S. Yan *et al.*, "Captum: A unified and generic model interpretability library for pytorch," *arXiv preprint arXiv:2009.07896*, 2020.
- [25] D. Kiela, H. Firooz, A. Mohan, V. Goswami, A. Singh, C. A. Fitzpatrick, P. Bull, G. Lipstein, T. Nelli, R. Zhu *et al.*, "The hateful memes challenge: Competition report," in *NeurIPS 2020 Competition and Demonstration Track*. PMLR, 2021, pp. 344–360.
- [26] U. Bhatt, J. Antorán, Y. Zhang, Q. V. Liao, P. Sattigeri, R. Fogliato, G. Melançon, R. Krishnan, J. Stanley, O. Tickoo *et al.*, "Uncertainty as a form of transparency: Measuring, communicating, and using uncertainty," in *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, 2021, pp. 401–413.
- [27] E. Hüllermeier and W. Waegeman, "Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods," *Machine Learning*, vol. 110, pp. 457–506, 2021.
- [28] O. Rahmati, B. Choubin, A. Fathabadi, F. Coulon, E. Soltani, H. Shahabi, E. Mollaefar, J. Tiefenbacher, S. Cipullo, B. B. Ahmad *et al.*, "Predicting uncertainty of machine learning models for modelling nitrate pollution of groundwater using quantile regression and unec methods," *Science of the Total Environment*, vol. 688, pp. 855–866, 2019.
- [29] P. Manakul, A. Liusie, and M. J. Gales, "Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models," *arXiv preprint arXiv:2303.08896*, 2023.
- [30] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," *Advances in Neural Information Processing Systems*, vol. 35, pp. 27 730–27 744, 2022.
- [31] X. Hou, Y. Zhao, Y. Liu, Z. Yang, K. Wang, L. Li, X. Luo, D. Lo, J. Grundy, and H. Wang, "Large language models for software engineering: A systematic literature review," 2023.
- [32] A. Fan, B. Gokkaya, M. Harman, M. Lyubarskiy, S. Sengupta, S. Yoo, and J. M. Zhang, "Large language models for software engineering: Survey and open problems," *arXiv preprint arXiv:2310.03533*, 2023.
- [33] D. Fried, A. Aghajanyan, J. Lin, S. Wang, E. Wallace, F. Shi, R. Zhong, S. Yih, L. Zettlemoyer, and M. Lewis, "InCoder: A generative model for code infilling and synthesis," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: <https://openreview.net/forum?id=hQwb-lbM6EL>
- [34] L. B. Allal, R. Li, D. Kocetkov, C. Mou, C. Akiki, C. M. Ferrandis, N. Muennighoff, M. Mishra, A. Gu, M. Dey *et al.*, "Santacoder: don't reach for the stars!" *arXiv preprint arXiv:2301.03988*, 2023.
- [35] E. Nijkamp, B. Pang, H. Hayashi, L. Tu, H. Wang, Y. Zhou, S. Savarese, and C. Xiong, "Codegen: An open large language model for code with multi-turn program synthesis," *arXiv preprint arXiv:2203.13474*, 2022.
- [36] Q. Gu, "Llm-based code generation method for golang compiler testing," in *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2023, pp. 2201–2203.
- [37] S. Kang, J. Yoon, and S. Yoo, "Large language models are few-shot testers: Exploring llm-based general bug reproduction," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023, pp. 2312–2323.
- [38] C. Lemieux, J. P. Inala, S. K. Lahiri, and S. Sen, "Codamosa: Escaping coverage plateaus in test generation with pre-trained large language models," in *International conference on software engineering (ICSE)*, 2023.
- [39] Z. Liu, C. Chen, J. Wang, X. Che, Y. Huang, J. Hu, and Q. Wang, "Fill in the blank: Context-aware automated text input generation for mobile gui testing," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023, pp. 1355–1367.
- [40] R. Tian, Y. Ye, Y. Qin, X. Cong, Y. Lin, Z. Liu, and M. Sun, "Debugbench: Evaluating debugging capability of large language models," 2024.
- [41] E. First, M. Rabe, T. Ringer, and Y. Brun, "Baldur: Whole-proof generation and repair with large language models," in *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2023, pp. 1229–1241.
- [42] Y. Wei, C. S. Xia, and L. Zhang, "Copiloting the copilots: Fusing large language models with completion engines for automated program repair," in *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2023, pp. 172–184.
- [43] M. Geng, S. Wang, D. Dong, H. Wang, G. Li, Z. Jin, X. Mao, and X. Liao, "Large language models are few-shot summarizers: Multi-intent comment generation via in-context learning," 2024.
- [44] X. Zhang, X. Xie, L. Ma, X. Du, Q. Hu, Y. Liu, J. Zhao, and M. Sun, "Towards characterizing adversarial defects of deep learning software from the lens of uncertainty," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 2020, pp. 739–751.
- [45] J. Kim, R. Feldt, and S. Yoo, "Guiding deep learning system testing using surprise adequacy," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019, pp. 1039–1049.
- [46] M. Weiss and P. Tonella, "Uncertainty quantification for deep neural networks: An empirical comparison and usage guidelines," *Software Testing, Verification and Reliability*, p. e1840, 2023.
- [47] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.
- [48] T. Kuhn, H. Niemann, and E. G. Schukat-Talamazzini, "Ergodic hidden markov models and polygrams for language modeling," in *Proceedings of ICASSP'94. IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 1. IEEE, 1994, pp. 1–357.
- [49] T. Brants, A. Popat, P. Xu, F. J. Och, and J. Dean, "Large language models in machine translation," in *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 2007, pp. 858–867.
- [50] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, "Recurrent neural network based language model," in *Interspeech*, vol. 2, no. 3. Makuhari, 2010, pp. 1045–1048.
- [51] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.
- [52] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," *arXiv preprint arXiv:1801.06146*, 2018.
- [53] Y. Liu and M. Lapata, "Text summarization with pretrained encoders," *arXiv preprint arXiv:1908.08345*, 2019.
- [54] J. Yang, M. Wang, H. Zhou, C. Zhao, W. Zhang, Y. Yu, and L. Li, "Towards making the most of bert in neural machine translation," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 05, 2020, pp. 9378–9385.
- [55] J. D. M.-W. C. Kenton and L. K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of NAACL-HLT*, 2019, pp. 4171–4186.
- [56] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.
- [57] D. Guo, S. Ren, S. Lu, Z. Feng, D. Tang, S. Liu, L. Zhou, N. Duan, A. Svyatkovskiy, S. Fu *et al.*, "Graphcodebert: Pre-training code representations with data flow," *arXiv preprint arXiv:2009.08366*, 2020.
- [58] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 7871–7880.
- [59] Y. Wang, W. Wang, S. Joty, and S. C. Hoi, "Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation," *arXiv preprint arXiv:2109.00859*, 2021.
- [60] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

- [61] Y. Dong, X. Jiang, Z. Jin, and G. Li, "Self-collaboration code generation via chatgpt," *arXiv preprint arXiv:2304.07590*, 2023.
- [62] C. S. Xia and L. Zhang, "Keep the conversation going: Fixing 162 out of 337 bugs for \$0.42 each using chatgpt," *arXiv preprint arXiv:2304.00385*, 2023.
- [63] K. R. Varshney and H. Alemzadeh, "On the safety of machine learning: Cyber-physical systems, decision sciences, and data products," *Big data*, vol. 5, no. 3, pp. 246–255, 2017.
- [64] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar, "The security of machine learning," *Machine Learning*, vol. 81, pp. 121–148, 2010.
- [65] S. Lo Piano, "Ethical principles in machine learning and artificial intelligence: cases from the field and possible ways forward," *Humanities and Social Sciences Communications*, vol. 7, no. 1, pp. 1–7, 2020.
- [66] J. Yang, K. Zhou, Y. Li, and Z. Liu, "Generalized out-of-distribution detection: A survey," *arXiv preprint arXiv:2110.11334*, 2021.
- [67] S. Rabanser, S. Günnemann, and Z. Lipton, "Failing loudly: An empirical study of methods for detecting dataset shift," *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [68] C. Ning and F. You, "Optimization under uncertainty in the era of big data and deep learning: When machine learning meets mathematical programming," *Computers & Chemical Engineering*, vol. 125, pp. 434–448, 2019.
- [69] A. Ashukha, A. Lyzhov, D. Molchanov, and D. Vetrov, "Pitfalls of in-domain uncertainty estimation and ensembling in deep learning," *arXiv preprint arXiv:2002.06470*, 2020.
- [70] Y.-C. Hsu, Y. Shen, H. Jin, and Z. Kira, "Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10951–10960.
- [71] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" *Advances in neural information processing systems*, vol. 30, 2017.
- [72] J. Gawlikowski, C. R. N. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, R. Roscher *et al.*, "A survey of uncertainty in deep neural networks," *arXiv preprint arXiv:2107.03342*, 2021.
- [73] P. Oberdiek, M. Rottmann, and H. Gottschalk, "Classification uncertainty of deep neural networks based on gradient information," in *Artificial Neural Networks in Pattern Recognition: 8th IAPR TC3 Workshop, ANNPR 2018, Siena, Italy, September 19–21, 2018, Proceedings 8*. Springer, 2018, pp. 113–125.
- [74] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," *Advances in neural information processing systems*, vol. 30, 2017.
- [75] D. Barber and C. M. Bishop, "Ensemble learning in bayesian neural networks," *Nato ASI Series F Computer and Systems Sciences*, vol. 168, pp. 215–238, 1998.
- [76] A. Lyzhov, Y. Molchanova, A. Ashukha, D. Molchanov, and D. Vetrov, "Greedy policy search: A simple baseline for learnable test-time augmentation," in *Conference on Uncertainty in Artificial Intelligence*. PMLR, 2020, pp. 1308–1317.
- [77] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.
- [78] G. Wang, W. Li, M. Aertsen, J. Deprest, S. Ourselin, and T. Vercauteren, "Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks," *Neurocomputing*, vol. 338, pp. 34–45, 2019.
- [79] X. Huang, W. Ruan, W. Huang, G. Jin, Y. Dong, C. Wu, S. Bensalem, R. Mu, Y. Qi, X. Zhao *et al.*, "A survey of safety and trustworthiness of large language models through the lens of verification and validation," *arXiv preprint arXiv:2305.11391*, 2023.
- [80] X. Chen, J. Ye, C. Zu, N. Xu, R. Zheng, M. Peng, J. Zhou, T. Gui, Q. Zhang, and X. Huang, "How robust is gpt-3.5 to predecessors? a comprehensive study on language understanding tasks," *arXiv preprint arXiv:2303.00293*, 2023.
- [81] M. Jang and T. Lukasiewicz, "Consistency analysis of chatgpt," *arXiv preprint arXiv:2303.06273*, 2023.
- [82] Y. Xiao and W. Y. Wang, "On hallucination and predictive uncertainty in conditional language generation," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, 2021, pp. 2734–2744.
- [83] A. Malinin and M. Gales, "Uncertainty estimation in autoregressive structured prediction," in *International Conference on Learning Representations*.
- [84] L. Kuhn, Y. Gal, and S. Farquhar, "Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation," *arXiv preprint arXiv:2302.09664*, 2023.
- [85] J. Alammar, "Ecco: An open source library for the explainability of transformer language models," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, 2021.
- [86] A. Galassi, M. Lippi, and P. Torroni, "Attention in natural language processing," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 10, pp. 4291–4308, 2020.
- [87] J. Vig and Y. Belinkov, "Analyzing the structure of attention in a transformer language model," in *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 2019, pp. 63–76.
- [88] "Gpt 3.5," <https://platform.openai.com/docs/models/gpt-3-5>, 2023.
- [89] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24–26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. [Online]. Available: <https://openreview.net/forum?id=Hkg4TI9xl>
- [90] U. Arora, W. Huang, and H. He, "Types of out-of-distribution texts and how to detect them," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 10687–10701.
- [91] Y. Gal, "Uncertainty in deep learning." University of Cambridge., 2016.
- [92] Z. Wang, Y. Huang, L. Ma, H. Yokoyama, S. Tokumoto, and K. Munakata, "An exploratory study of ai system risk assessment from the lens of data distribution and uncertainty," *arXiv preprint arXiv:2212.06828*, 2022.
- [93] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.
- [94] C. Meister, T. Pimentel, G. Wiher, and R. Cotterell, "Locally typical sampling," *Transactions of the Association for Computational Linguistics*, vol. 11, pp. 102–121, 2023.
- [95] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan *et al.*, "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024.
- [96] G. Team, M. Riviere, S. Pathak, P. G. Sessa, C. Hardin, S. Bhupatiraju, L. Hussenot, T. Mesnard, B. Shahriari, A. Ramé *et al.*, "Gemma 2: Improving open language models at a practical size," *arXiv preprint arXiv:2408.00118*, 2024.
- [97] M. Abdin, S. A. Jacobs, A. A. Awan, J. Aneja, A. Awadallah, H. Awadalla, N. Bach, A. Bahree, A. Bakhtiari, H. Behl *et al.*, "Phi-3 technical report: A highly capable language model locally on your phone," *arXiv preprint arXiv:2404.14219*, 2024.
- [98] J. Bai, S. Bai, Y. Chu, Z. Cui, K. Dang, X. Deng, Y. Fan, W. Ge, Y. Han, F. Huang, B. Hui, L. Ji, M. Li, J. Lin, R. Lin, D. Liu, G. Liu, C. Lu, K. Lu, J. Ma, R. Men, X. Ren, X. Ren, C. Tan, S. Tan, J. Tu, P. Wang, S. Wang, W. Wang, S. Wu, B. Xu, J. Xu, A. Yang, H. Yang, J. Yang, S. Yang, Y. Yao, B. Yu, H. Yuan, Z. Yuan, J. Zhang, X. Zhang, Y. Zhang, Z. Zhang, C. Zhou, J. Zhou, X. Zhou, and T. Zhu, "Qwen technical report," *arXiv preprint arXiv:2309.16609*, 2023.
- [99] D. Guo, Q. Zhu, D. Yang, Z. Xie, K. Dong, W. Zhang, G. Chen, X. Bi, Y. Wu, Y. Li *et al.*, "Deepseek-coder: When the large language model meets programming—the rise of code intelligence," *arXiv preprint arXiv:2401.14196*, 2024.
- [100] A. Lozhkov, R. Li, L. B. Allal, F. Cassano, J. Lamy-Poirier, N. Tazi, A. Tang, D. Pykhtar, J. Liu, Y. Wei *et al.*, "StarCoder 2 and the stack v2: The next generation," *arXiv preprint arXiv:2402.19173*, 2024.
- [101] "Gpt-4o," <https://openai.com/index/hello-gpt-4o/>, 2024.
- [102] "Gpt-4o mini," <https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/>, 2024.
- [103] Y. Yang, W.-t. Yih, and C. Meek, "WikiQA: A challenge dataset for open-domain question answering," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 2013–2018. [Online]. Available: <https://aclanthology.org/D15-1237>
- [104] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*. Barcelona, Spain:

- Association for Computational Linguistics, Jul. 2004, pp. 74–81. [Online]. Available: <https://aclanthology.org/W04-1013>
- [105] S. Ren, D. Guo, S. Lu, L. Zhou, S. Liu, D. Tang, N. Sundaresan, M. Zhou, A. Blanco, and S. Ma, “Codebleu: a method for automatic evaluation of code synthesis,” *arXiv preprint arXiv:2009.10297*, 2020.
- [106] J. Gao, Q. Zhou, and R. Qiu, “ELI5-Category: a categorized open-domain qa dataset,” 2021.
- [107] R. Nallapati, B. Zhou, C. Gulcehre, B. Xiang *et al.*, “Abstractive text summarization using sequence-to-sequence rnns and beyond,” *arXiv preprint arXiv:1602.06023*, 2016.
- [108] O. Bojar, C. Buck, C. Federmann, B. Haddow, P. Koehn, J. Leveling, C. Monz, P. Pecina, M. Post, H. Saint-Amand, R. Soricut, L. Specia, and A. Tamchyna, “Findings of the 2014 workshop on statistical machine translation,” in *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Baltimore, Maryland, USA: Association for Computational Linguistics, Jun. 2014, pp. 12–58. [Online]. Available: <https://aclanthology.org/W14-3302>
- [109] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman *et al.*, “Evaluating large language models trained on code,” *arXiv preprint arXiv:2107.03374*, 2021.
- [110] J. Austin, A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. Cai, M. Terry, Q. Le *et al.*, “Program synthesis with large language models,” *arXiv preprint arXiv:2108.07732*, 2021.
- [111] N. Reimers and I. Gurevych, “Sentence-bert: Sentence embeddings using siamese bert-networks,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. [Online]. Available: <https://arxiv.org/abs/1908.10084>
- [112] X. Chen, M. Lin, N. Schärli, and D. Zhou, “Teaching large language models to self-debug,” *arXiv preprint arXiv:2304.05128*, 2023.
- [113] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano *et al.*, “Training verifiers to solve math word problems,” *arXiv preprint arXiv:2110.14168*, 2021.
- [114] “all-mpnet-base-v2,” <https://huggingface.co/sentence-transformers/all-mpnet-base-v2>, 2023.
- [115] S. Arora and A. Goyal, “A theory for emergence of complex skills in language models,” *arXiv preprint arXiv:2307.15936*, 2023.
- [116] M. T. Ribeiro, T. Wu, C. Guestrin, and S. Singh, “Beyond accuracy: Behavioral testing of NLP models with CheckList,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Eds. Online: Association for Computational Linguistics, Jul. 2020, pp. 4902–4912. [Online]. Available: <https://aclanthology.org/2020.acl-main.442>
- [117] J. Ferrando, M. Sperber, H. Setiawan, D. Telaar, and S. Hasan, “Automating behavioral testing in machine translation,” in *Proceedings of the Eighth Conference on Machine Translation*, P. Koehn, B. Haddow, T. Kocmi, and C. Monz, Eds. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 1014–1030. [Online]. Available: <https://aclanthology.org/2023.wmt-1.97>
- [118] J. Lee, S. Chen, A. Mordahl, C. Liu, W. Yang, and S. Wei, “Automated testing linguistic capabilities of nlp models,” *ACM Transactions on Software Engineering and Methodology*, 2024.
- [119] S. Li, X. Ning, L. Wang, T. Liu, X. Shi, S. Yan, G. Dai, H. Yang, and Y. Wang, “Evaluating quantized large language models,” *arXiv preprint arXiv:2402.18158*, 2024.
- [120] C. S. Xia, Y. Wei, and L. Zhang, “Automated program repair in the era of large pre-trained language models,” in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023, pp. 1482–1494.
- [121] R. Pan, A. R. Ibrahimzada, R. Krishna, D. Sankar, L. P. Wassi, M. Merler, B. Sobolev, R. Pavuluri, S. Sinha, and R. Jabbarvand, “Lost in translation: A study of bugs introduced by large language models while translating code,” in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE ’24. New York, NY, USA: Association for Computing Machinery, 2024. [Online]. Available: <https://doi-org.login.ezproxy.library.ualberta.ca/10.1145/3597503.3639226>



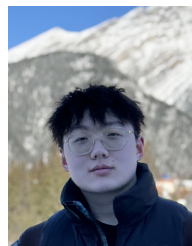
**Yuheng Huang** (Graduate Student Member, IEEE) received a BE degree from the Beijing University of Posts and Telecommunications, China, and a master’s degree from the University of Alberta, Canada. He is currently working toward a PhD degree at The University of Tokyo, Japan. His research interests primarily focus on the quality assurance of AI-enabled complex systems as well as developing interactive interfaces that can enable more efficient human-in-the-loop AI development from Human-Computer Interaction (HCI) perspectives.



**Jiayang Song** (Graduate Student Member, IEEE) received the B.E. degree from Western University, London, ON, Canada, in 2019 and the M.E. degree from the University of Toronto, Toronto, ON, Canada, in 2021. He is currently pursuing a Ph.D. degree with the University of Alberta, Edmonton, AB, Canada. His research interests include testing, analysis, repairing, and enhancement of AI systems and their applications for quality assurance of trustworthy AI-enabled cyber-physical systems.



**Zhijie Wang** (Graduate Student Member, IEEE) is currently a Ph.D. candidate in Software Engineering at the University of Alberta, AB, Canada. Previously, he received his M.E. degree from the University of Waterloo, ON, Canada in 2021. His research interest focuses on software engineering support for complex AI-based software systems. He is also broadly interested in the intersection of software engineering, AI, and human-computer interaction (HCI). His work has been published in top-tier SE and HCI venues and has received a best paper award (FSE ’23).



**Shengming Zhao** received a B.E. degree from the Beijing University of Posts and Telecommunications, Beijing, China. He is currently pursuing a M.Sc. degree with the University of Alberta, Edmonton, Canada. His research interests include quality assurance of AI systems and automatic code generation.



**Huaming Chen** (Member, IEEE) received the Ph.D. degree from the University of Wollongong, Wollongong, Australia. He is currently a senior lecturer with the School of Electrical and Computer Engineering, the University of Sydney, Sydney, Australia. His main research interests include software engineering/security, trustworthy AI, and applied machine learning. He regularly serves on the program committees of ACM MM, CCS, ACSAC, SANER, IJCAI, KDD, The Web Conference, SIAM ICDM, ECML/PKDD and so on.



**Felix Juefei-Xu** (Member, IEEE) received the Ph.D. degree in Electrical and Computer Engineering from Carnegie Mellon University (CMU), Pittsburgh, PA, USA. Prior to that, he received the M.S. degree in Electrical and Computer Engineering and the M.S. degree in Machine Learning from CMU, and the B.S. degree in Electronic Engineering from Shanghai Jiao Tong University (SJTU), Shanghai, China. Currently, he is a Research Scientist with GenAI at Meta, based in New York City, where he works on robust perception and efficient learning problems in

the domain of generative AI. He is also affiliated with New York University as an Adjunct Professor. Previously, he was a Research Scientist with Alibaba Group, based in Sunnyvale, CA. He was the recipient of multiple best or distinguished paper awards, including IJCB 2011, BTAS 2015 and 2016, ASE 2018, and ACCV 2018.



**Lei Ma** (Member, IEEE) received the B.E. degree from Shanghai Jiao Tong University, Shanghai, China, in 2009, and the M.E. and Ph.D. degrees from The University of Tokyo, Tokyo, Japan, in 2011 and 2014, respectively. He is currently an Associate Professor with The University of Tokyo and the University of Alberta, Edmonton, AB, Canada. He was honorably selected as Canada CIFAR AI Chair and a fellow with Alberta Machine Intelligence Institute (Amii), Edmonton. His research interests include the interdisciplinary fields of software engineering (SE)

and trustworthy artificial intelligence, with a special focus on the quality, reliability, safety, and security aspects of AI systems.