Decomposing Uncertainty for Large Language Models through Input Clarification Ensembling

Bairu Hou [1] **Yujian Liu** [1] **Kaizhi Qian** [2] **Jacob Andreas** [3] **Shiyu Chang** [1] **Yang Zhang** [2]

Abstract

Uncertainty decomposition refers to the task of decomposing the total uncertainty of a predictive model into aleatoric (data) uncertainty, resulting from inherent randomness in the data-generating process, and epistemic (model) uncertainty, resulting from missing information in the model's training data. In large language models (LLMs) specifically, identifying sources of uncertainty is an important step toward improving reliability, trustworthiness, and interpretability, but remains an important open research question. In this paper, we introduce an uncertainty decomposition framework for LLMs, called input clarification ensembling, which can be applied to any pre-trained LLM. Our approach generates a set of clarifications for the input, feeds them into an LLM, and ensembles the corresponding predictions. We show that, when aleatoric uncertainty arises from ambiguity or under-specification in LLM inputs, this approach makes it possible to factor an (un-clarified) LLM's predictions into separate aleatoric and epistemic terms, using a decomposition similar to the one employed by Bayesian neural networks. Empirical evaluations demonstrate that input clarification ensembling provides accurate and reliable uncertainty quantification on several language processing tasks. Code and data are available at https://github.com/UCSB-NLP-Chang/llm_uncertainty.

1. Introduction

With the widespread application of large language models (LLMs), it is becoming crucial to ensure predictions from LLMs are trustworthy. One critical dimension of trustworthiness is the ability to indicate when generated text is reliable and correct, which may be formalized as the problem of *uncertainty quantification* (UQ). Uncertainty quantification aims to measure the confidence level of neural networks in their predictions (Gal et al., 2016; Bhatt et al., 2021; Hüllermeier & Waegeman, 2021). A higher uncertainty implies the output of LLMs should be clarified, manually evaluated, or rejected.

Quantifying LLMs' *total uncertainty* has been the focus of increasing research attention. Existing work observes that LLMs are relatively well-calibrated, especially when predictions are obtained by ensembling multiple reasoning chains (Wang et al., 2022; Huang et al., 2022; Si et al., 2023) or prompts (Jiang et al., 2023), or when LLMs are prompted to directly output their confidence levels (Kadavath et al., 2022; Lin et al., 2022; Tian et al., 2023). Many other methods have been proposed to quantify the uncertainty of LLMs (Lin et al., 2022; Xiao et al., 2022; Kuhn et al., 2022; Lin et al., 2023; Duan et al., 2023; Huang et al., 2023; Park & Kim, 2023; Ren et al., 2023). Accurate quantification of the uncertainty can be used for various applications, such as out-of-distribution detection and misclassified data detection.

However, *measuring* uncertainty is just the first step towards *understanding* uncertainty in LLM predictions. For many applications, it is necessary to distinguish between different types of uncertainty and decompose the source into these types, a problem we refer to as *uncertainty decomposition*. As discussed more formally below, it is always possible to decompose a predictive model's uncertainty into two components: *aleatoric (data) uncertainty* and *epistemic (model) uncertainty*. Epistemic uncertainty arises when correct outputs are predictable in principle, but models lack the knowledge required for prediction. For example, the question *What is 2+3?* requires the knowledge of algebraic operations. Without such knowledge, the uncertainty will be high. On the other hand, aleatoric uncertainty arises from ambiguity or inherent randomness in the data-generating process itself: in language processing applications, it may result from ambiguous questions (Min et al., 2020; Guo et al., 2021; Kuhn et al., 2023) and unclear task instructions (Tamkin et al., 2022). In particular, an important source of aleatoric uncertainty is the *input ambiguity*. For example, the answer to

[1]UC Santa Barbara [2]MIT-IBM Watson AI Lab, IBM Research [3]MIT CSAIL. Correspondence to: Bairu Hou <bairu@ucsb.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

the input question *Who is the president of this* have a high aleatoric uncertainty because it i what country and time the question intends to

This paper aims to obtain a finer-grained unc surement by determining how much of the tot can be attributed to aleatoric uncertainty due biguity. Aleatoric uncertainty due to input irreducible no matter how well a model learns. to answer the question *Who is the president try?*, without any context, the uncertainty w regardless of how well the LLM learns, becaus itself is ambiguous. Uncertainty decomposit important insights for users to improve the pe LLM. If epistemic uncertainty is high, users the model with adequate knowledge through r tion, in-context learning, *etc.*; if the aleatoric u high, then users should modify the query to make it more concrete.

Despite existing work aimed at quantifying total uncertainty in LLMs, decomposing this uncertainty for LLMs remains understudied. Existing methods for uncertainty decomposition in other models cannot be directly applied, due to the black-box nature of LLMs and the prohibitive cost of inference. For example, Bayesian Neural Networks (BNNs) (Neal, 2012; Blundell et al., 2015; Graves, 2011; Louizos & Welling, 2016; Hernández-Lobato & Adams, 2015; Hasenclever et al., 2017; Li et al., 2015) specify a prior distribution over the model parameters and approximate the posterior distribution given the training data. DEEP ENSEMBLES (Lakshminarayanan et al., 2017; Fort et al., 2019) decompose the uncertainty by training different variants of models, *e.g.*, with different random seeds, to with the proper scoring rules (*e.g.*, negative log-likelihood loss for the classification task) in the target task and then ensembling them.

Despite their effectiveness, Bayesian Neural Networks require substantial modifications to the training procedure, while DEEP ENSEMBLES necessitate training multiple variants of LLMs. Both approaches are generally impractical or prohibitively expensive. Given these challenges, we aim to address the following question: *How can we effectively quantify and decompose uncertainty in LLMs?*

In this paper, we propose a framework for uncertainty decomposition that we call *input clarification ensembling*. Our approach shares many intuitions and structural similarities with BNN-based approaches, but avoid the need to modify LLM parameters or inference procedures. Our approach is motivated by the observation that, although it is very challenging to modify LLM's parameters, it is relatively easy to manipulate the input to LLMs. Inspired by this, rather than ensembling different model variants that minimize the epistemic uncertainty, we introduce a set of *input clarifications* which can minimize the aleatoric uncertainty. We then ensemble an LLM's predictions under different clarifications. Figure 1 shows the general pipeline. For example, for the question *Who is the president of this country?*, a possible clarification is *'This country' refers to the US.* By ruling out the aleatoric uncertainty by clarification, we can ascribe the remaining uncertainty of each individual prediction to epistemic uncertainty. Furthermore, by measuring the disagreement of the model predictions under different clarifications, we can gauge the aleatoric uncertainty. Our experiments verify that the proposed method provide accurate uncertainty quantification results on both total uncertainty and its decomposition.

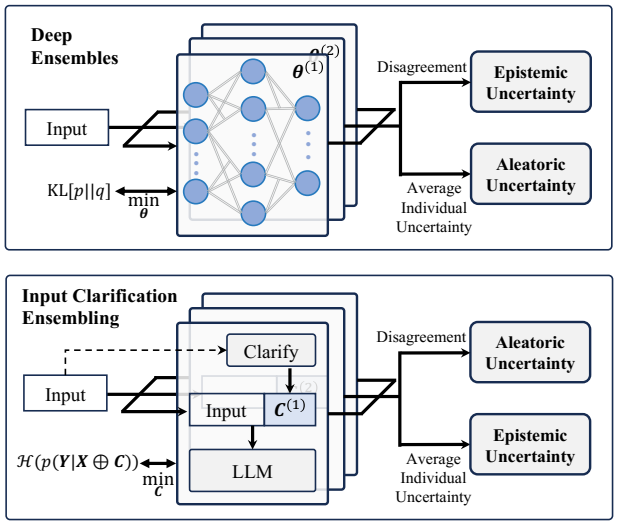

Figure 1. The uncertainty quantification frameworks of DEEP ENSEMBLES (upper) and input clarification ensembling (lower).

2. Related Work

Uncertainty quantification. Uncertainty quantification for machine learning models has been widely studied to quantify the reliability of model predictions (Gal et al., 2016; Gal & Ghahramani, 2016; Malinin & Gales, 2018; Ovadia et al., 2019; Malinin et al., 2020; Lin et al., 2022; Kuhn et al., 2022; Lin et al., 2023; Shen et al., 2023). Uncertainty in model predictions can have numerous causes. Given the total uncertainty in model predictions, one can further decompose it into epistemic uncertainty (due to lack of knowledge in the model) and aleatoric uncertainty (due to the inherent randomness and noise in data).

Depending on how the uncertainty is obtained, existing uncertainty quantification methods can be categorized into *intrinsic* and *extrinsic* methods. Intrinsic methods adopt machine learning models to provide an inherent uncertainty estimate, such as Bayesian approaches and ensemble-based approaches (Malinin & Gales, 2018). Bayesian approaches (Blundell et al., 2015; Gal & Ghahramani, 2016;

Teye et al., 2018; Mobiny et al., 2021; Lakshminarayanan et al., 2017; Malinin et al., 2020; He et al., 2020) and ensemble-based approaches (Lakshminarayanan et al., 2017; Fort et al., 2019) can quantify both aleatoric and epistemic uncertainty. In comparison, extrinsic methods quantify the uncertainty in a post-hoc manner using auxiliary models (Kristiadi et al., 2021; Lahlou et al., 2022). Our method belongs to the intrinsic family of methods and is directly motivated by Bayesian neural network approaches.

Uncertainty Quantification and Model Calibration for LLMs With the wide application of LLMs, how to accurately quantify the predictive uncertainty has also drawn attention (Xiao et al., 2022; Lin et al., 2022; Mielke et al., 2022; Zhou et al., 2023; Huang et al., 2023; Duan et al., 2023; Chen & Mueller, 2023; Ott et al., 2018; Malinin & Gales, 2020). Semantic Uncertainty (Kuhn et al., 2022) clusters string-valued LLM outputs by synonymy for better uncertainty quantification. Lin et al. (2023) explores uncertainty quantification within the challenging black-box context, where the token generation probability is inaccessible. In this pursuit, BSDETECTOR (Chen & Mueller, 2023) combines two strategies to estimate the model's predictive uncertainty. The first approach involves sampling multiple answers from the LLM and assessing their consistency, while the second directly queries the LLM for its confidence in the generated answer. Although there have been some explorations in this direction, existing methods can only estimate the total uncertainty. In comparison, we propose a more principled framework that can both quantify the total uncertainty and decompose it into aleatoric uncertainty and epistemic uncertainty, leading to a more fine-grained understanding of LLMs.

Another line of research is model calibration for LLMs. Model calibration is the process of ensuring that the predicted probabilities or confidence scores from a machine learning model align with the true probabilities or likelihoods of events occurring (*i.e.*, the prediction is correct). Well-calibrated model predictions help improve the reliability of uncertainty quantification. Using existing model calibration methods (Hendrycks & Gimpel, 2016; Guo et al., 2017; Ovadia et al., 2019; Riquelme et al., 2018; Desai & Durrett, 2020), prior work (Huang et al., 2022; Jiang et al., 2023; 2021; Ye & Durrett, 2022) has shown that LLMs are relatively well-calibrated on factual QA and complex reasoning tasks when properly prompted. Specifically, Kadavath et al. (2022); Tian et al. (2023) estimate the prediction confidence of LLMs by prompting LLMs to output their confidence of their answers. For complex reasoning tasks, LLMs may output both the reasoning chains and the final answer. To estimate the confidence score, previous approaches (Huang et al., 2022) sample multiple outputs for the input question and use the answer frequency to indicate the confidence. Researchers further ensemble multiple prompts for better calibration performance (Jiang et al., 2023). Our uncertainty quantification is based on the well-calibrated predictions of LLMs, which lead to a more precise and accurate quantification result.

Modeling Ambiguity with language models Ambiguity is a longstanding issue in the NLP domain, extensively explored in tasks such as syntactic and semantic parsing (Koller et al., 2008), open-domain question-answering (Min et al., 2020; Cole et al., 2023), conversational question-answering (Guo et al., 2021) and natural language inference (Liu et al., 2023). Prior work, such as `AmbigQA` (Min et al., 2020) and `AmbigEnt` (Liu et al., 2023), have identified the widespread ambiguities and established benchmarks with ambiguous inputs. These studies have demonstrated that existing language models lack the capability to effectively recognize and manage ambiguities. Our work models the ambiguity from the perspective of uncertainty quantification, where a high aleatoric uncertainty can indicate the potential existence of input ambiguity. By decomposing the aleatoric uncertainty, we show that it is possible to enhance the ambiguity detection performance of existing LLMs.

3. Methodology

3.1. Notations and Problem Formulation

Denote by $\boldsymbol{X}$ and $\boldsymbol{Y}$ the input and output target of a given task and $\boldsymbol{\theta}$ as the parameters of an LLM. Denote by $p(\boldsymbol{Y}|\boldsymbol{X})$ and $q(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\theta})$ the ground-truth and predicted distribution of $\boldsymbol{Y}$ given $\boldsymbol{X}$.

We first introduce three uncertainty concepts. First, the ***total uncertainty*** is defined as the entropy of the predicted distribution, *i.e.*, $\mathcal{U}_{total} = \mathcal{H}(q(\boldsymbol{Y}|\boldsymbol{X}; \boldsymbol{\theta}))$. If the overall uncertainty is high, then it means the LLM has low confidence in its output. The total uncertainty can be further decomposed into two different types of uncertainties.

The first type of uncertainty is referred to as the ***epistemic uncertainty***, which characterizes how well the LLM approaches the ground truth distribution, and thus learns the knowledge therein. For example, to answer *'What is 2+3?'*, if the LLM were able to learn the true knowledge of the algebraic operation, it would be able to answer with certainty; otherwise, the uncertainty would be high.

The second type of uncertainty is referred to as the ***aleatoric uncertainty***, which characterizes the fundamental uncertainty residing in the ground-truth distribution, and is irreducible no matter how well the LLM learns. For example, to answer *'Who is the president of this country?'*, even if the LLM were well acquainted with politics, it still could not answer it confidently, because this question is inherently

ambiguous. The data aleatoric is often quantified by the entropy in the ground-truth distribution, *i.e.*, $\mathcal{H}(p(\boldsymbol{Y}|\boldsymbol{X}))$.

The goal of this paper is to estimate both the epistemic and aleatoric uncertainties in LLMs.

3.2. Background: Bayesian Neural Networks and DEEP ENSEMBLES

The possible solutions to our task is to apply the canonical Bayesian Neural Network (BNN) approach (Blundell et al., 2015; Graves, 2011) or DEEP ENSEMBLES (Lakshminarayanan et al., 2017), which are standard approaches to uncertainty decomposition. Instead of having one set of parameters, BNNs model the parameter distribution of a neural network. With the Bayesian formalism, the posterior distribution can be approximated give the training data via techniques such as variational inference (Blundell et al., 2015; Graves, 2011). Due to the prohibitive computational cost of BNNs, non-Bayesian methods such as DEEP ENSEMBLES are proposed with better scalability. DEEP ENSEMBLES maintain K models, each parameterized as $\boldsymbol{\theta}^{(k)}$. Each of the k models seeks to minimize the training loss, usually the cross entropy loss for classification tasks, which is equivalent to solving the following optimization problem

$$\min_{\boldsymbol{\theta}} \mathrm{KL}(p(\boldsymbol{Y}|\boldsymbol{X})\|q(\boldsymbol{Y}|\boldsymbol{X},\boldsymbol{\theta}))). \tag{1}$$

In DEEP ENSEMBLES, different models have slightly different initialization values and thus the optimized values, $\{\boldsymbol{\theta}^{(k)}\}$, are different. Denote the resulting distribution of the model parameters θ as $p(\boldsymbol{\theta}|\mathcal{D})$ (either approximated by BNNs or DEEP ENSEMBLES) where $\mathcal{D}$ is the training dataset. Then the ensembled distribution of BNN can be represented as $q(\boldsymbol{Y}|\boldsymbol{X}) = \mathbb{E}_{q(\boldsymbol{\theta}|\mathcal{D})}[q(\boldsymbol{Y}|\boldsymbol{X},\boldsymbol{\theta})]$. Then we can decompose the predictive uncertainty as

$$\mathcal{H}(q(\boldsymbol{Y}|\boldsymbol{X})) = \underbrace{\mathcal{I}(\boldsymbol{Y};\boldsymbol{\theta}|\boldsymbol{X})}_{①} + \underbrace{\mathbb{E}_{q(\boldsymbol{\theta}|\mathcal{D})}\mathcal{H}(q(\boldsymbol{Y}|\boldsymbol{X},\boldsymbol{\theta}))}_{②}, \tag{2}$$

where $\mathcal{I}$ denotes the mutual information under the q distribution. ① measures the disagreement among the different models; ② measures the average uncertainty of each individual model. The above equation can be straightforwardly derived from the definition of conditional mutual information. Under certain assumptions, ① and ② can approximate the epistemic and aleatoric uncertainties, respectively (Gal et al., 2016). An illustration of the DEEP ENSEMBLES framework is shown in the upper panel of Figure 1.

Here is an intuitive explanation of why this is the case. According to Eq. 1, the goal of each model is to approach the ground-truth distribution, and thus can be viewed as the process of reducing the epistemic uncertainty. Therefore, if the optimization is successful, all the models will learn the true distribution, *i.e.*, $q(\boldsymbol{Y}|\boldsymbol{X},\boldsymbol{\theta}^{(k)}) = p(\boldsymbol{Y}|\boldsymbol{X}), \forall k$, which, by definition, results in zero epistemic uncertainty. Meanwhile, ① will also be zero because all the models produce the same prediction. Thus ① equals epistemic uncertainty in this case. ② would also equal the aleatoric uncertainty because the predicted distribution is equal to the true distribution.

On the other hand, if the models fail to learn the true distribution, in which case the epistemic uncertainty will be large, ① will also be large since different models have different hyperparameter settings and will be stuck in very different local optima.

3.3. Do BNN and DEEP ENSEMBLES work for LLMs?

Our goal of decomposing uncertainty for LLMs would be easily achieved if these methods were readily applicable to LLMs. Unfortunately, this is not the case. For BNNs, we need to significantly modify the training method of LLMs. For DEEP ENSEMBLES, the learning process in Eq. 1 is also very challenging for LLMs. Specifically, there are two types of methods for adapting LLMs to a particular task, supervised fine-tuning and prompting/in-context learning. Directly fine-tuning the model according to Eq. 1 is usually infeasible due to the limited access to model parameters and its huge requirement for computation. Even if it is feasible, it would be very time-consuming because it requires fine-tuning multiple LLMs.

On the other hand, the in-context learning method, though feasible, does not fit into the DEEP ENSEMBLES framework because it does not directly aim to optimize Eq. 1, so the decomposition will be very inaccurate. To demonstrate this, we perform a simple experiment on the `AmbigQA` (Min et al., 2020) dataset, which contains both ambiguous questions with multiple answers and unambiguous questions. We use the BNN method to decompose the uncertainty of ChatGPT, where the different individual model is derived by providing different in-context examples. If the decomposition method is accurate, we would expect to see that the aleatoric uncertainty for the ambiguous questions is significantly larger than that of the unambiguous ones. However, as shown in Figure 2, the gap between the uncertainties of the two groups of questions is very small. More experiment details can be found in Section 4.

While the BNN and DEEP ENSEMBLE framework do not work for LLMs, it inspires us to design an alternative framework that is almost completely symmetrical to the BNN approach, as discussed in the next subsection.

3.4. Input Clarification Ensembling

Although modifying or adapting LLMs is challenging, it is relatively straightforward to modify the input to LLMs. By analogy to the way BNNs and DEEP ENSEMBLES ensemble

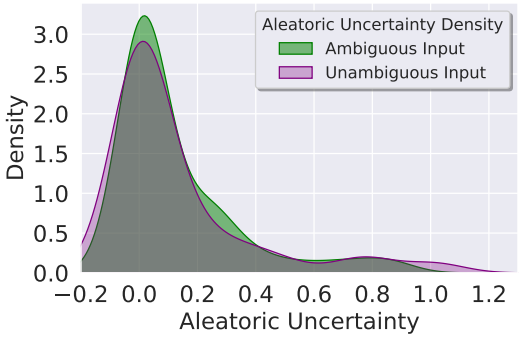

Figure 2. Aleatoric uncertainty distribution on the `AmbigQA` (Min et al., 2020) dataset using the DEEP ENSEMBLES method. We use kernel density estimation to smooth the frequency distribution histogram. DEEP ENSEMBLES is achieved by ensembling different in-context examples.

different *models* that minimize *epistemic uncertainty* (Eq. 1), can we design a framework that ensembles different *inputs* to minimize *aleatoric uncertainty*?

This is the motivation behind our framework, which consists of the following two steps.

Step 1: Input Clarification. Given an input $\boldsymbol{X}$, we first generate a set of texts, $\boldsymbol{C}^{(k)}$, called *clarifications*. Each clarification $\boldsymbol{C}^{(k)}$ seeks to minimize the ambiguity in $\boldsymbol{X}$ (and thus the aleatoric uncertainty) when appended to $\boldsymbol{X}$. Formally, we denote one clarification result as $\boldsymbol{X} \oplus \boldsymbol{C}^k$, where $\oplus$ denotes concatenation. In the aforementioned example, *'Who is the president of this country?'*, possible clarifications include *'This country refers to the US.'* and many other countries. Since there can be many valid clarifications for the input, $\{\boldsymbol{C}^{(k)}\}$ is a set.

Step 2: Ensemble. We denote the distribution of the aforementioned input clarifications as $q(\boldsymbol{C}|\boldsymbol{X})$ given a particular input $\boldsymbol{X}$. Then, we define the predictive model $q(\boldsymbol{Y}|\boldsymbol{X})$ as an ensemble of predictions conditional on each clarified input, *i.e.*, $q(\boldsymbol{Y}|\boldsymbol{X}) = \mathbb{E}_{q(\boldsymbol{C}|\boldsymbol{X})}[q(\boldsymbol{Y}|\boldsymbol{X} \oplus \boldsymbol{C}, \boldsymbol{\theta})]$. (Model parameters $\boldsymbol{\theta}$ are kept constant, and thus will be omitted for brevity below.)

We then propose to decompose the uncertainty of the ensembled model as

$$\mathcal{H}(q(\boldsymbol{Y}|\boldsymbol{X})) = \underbrace{\mathcal{I}(\boldsymbol{Y}; \boldsymbol{C}|\boldsymbol{X})}_{①'} + \underbrace{\mathbb{E}_{q(\boldsymbol{C}|\boldsymbol{X})}\mathcal{H}(q(\boldsymbol{Y}|\boldsymbol{X} \oplus \boldsymbol{C}))}_{②'}. \tag{3}$$

We claim that ①′, which computes the mutual information between the model output distribution and the clarifications, can approximate the aleatoric uncertainty caused by input ambiguity. In contrast, ②′ is the average entropy of the output distribution given different clarifications, representing the model's uncertainty across clarified versions of the input. Assuming that input ambiguity is the sole source of aleatoric uncertainty, we may consider it as an estimate of the epistemic uncertainty for the LLM given the original input. This interpretation, however, diverges from the traditional definition of epistemic uncertainty, and we mainly focus on decomposing the aleatoric uncertainty (①′) in this paper.

By comparing the above process against Eqs. 1 and 2, we can notice the symmetry between our framework and DEEP ENSEMBLES's — DEEP ENSEMBLES seeks to pin down epistemic uncertainty whereas ours aleatoric uncertainty; Eq. 3 takes almost an identical form to Eq. 2 but the corresponding uncertainties are swapped. Figure 1 also shows such symmetry.

Accordingly, the same explanation of why it works applies here. When the input is already very clear, and hence aleatoric uncertainty is low, the clarifications will be identically empty, so ①′ will approach zero. When the input is very ambiguous, the clarifications will be very different (think about the aforementioned president example), and so would the answers produced with different clarifications. In this case, ①′ will be very high. On the other hand, ②′ measures the average uncertainty on *clarified* input, which rules out most of the aleatoric uncertainty, so the remaining uncertainty can mostly be ascribed to epistemic uncertainty.

3.5. Input Clarification

Unlike the conventional neural networks, the input to LLMs usually contains multiple components, including instructions, in-context examples, questions *etc*. Therefore, we can separately measure the aleatoric uncertainties caused by different input components by clarifying only the corresponding components. For example, to measure the aleatoric uncertainty resulting from ambiguous instructions, we can clarify only the instruction. For the aleatoric uncertainty studied in this work, we will focus on the uncertainty caused by instruction ambiguity and question ambiguity, but the framework is readily generalizable to other input components.

To derive clarifications that approximately minimize the ambiguity in step 1 above, we introduce a clarification LLM, where we provide an instruction and in-context example to guide the LLM to perform adequate clarification. Therefore, the above input clarification distribution $q(\boldsymbol{C}|\boldsymbol{X})$ in Equation 3 is the output distribution of the clarification LLM. Note that the clarification LLM can be different from the LLM for prediction. In this work, we propose the following design choices for the clarification LLM to to ensure the quality of clarification:

- Prompting an LLM with task instructions and in-context examples. We design instructions for the clarification generation task and provide the model (`gpt-3.5-turbo` and `gpt-4`) with several in-context examples.

- Supervised fine-tuning. We can also fine-tune a open-sourced language model on datasets that contains the ambiguous inputs and their corresponding clarifications. We fine-tune the `Llama-3-8b-instruct` model on the training set of the `AmbigQA` (Min et al., 2020) dataset.

Further implementation details are provided in Section 4 and Appendix A.2. For both design choices, we show that we can easily adapt the LLMs for clarification generation and quantify the uncertainty using our proposed framework.

3.6. Improving Performance via Soliciting Clarifications

Our framework not only provides a way of decomposing the uncertainties, but can also enable an interpretable and effective human-LLM interaction experience. Currently, one of the major ways for humans to interact with LLMs is designing appropriate input. However, the input designed by humans may not be clear enough to LLMs, often resulting in undesirable answers given by LLMs. With the proposed input clarification framework, we can design an interaction paradigm that alleviates this problem.

Given an input query, we can first gauge the uncertainties of different input components. If one of the components, say the instruction, contributes to high uncertainty (exceeding a threshold), we can provide feedback to the user that the LLM is not sure about the answer because the instruction is ambiguous, along with several clarification options produced by the clarification LLM for the user to choose from. This would help the user to perform directed improvement of the input query and obtain the desirable answer.

4. Experiments

In this section, we conduct empirical evaluations to demonstrate the validity and effectiveness of the proposed method. Specifically, we aims to answer the following two questions:

1. Can the proposed UQ framework quantify *total uncertainty* effectively and correctly?
2. Can the proposed UQ framework *decompose the uncertainty* effectively and correctly?

To answer the first question, we conduct the mistake detection experiment, which will be introduced in Section 4.2. To answer the second question, we conduct three experiments: ambiguity detection, monotonicity check, and recall of correct answers, which will be presented in Sections 4.3-4.5, respectively.

4.1. Experiment Configurations

We use `gpt-3.5-turbo-0613` as the default LLM for all experiments. We sample 10 model predictions with temperature `0.5` and use the answer frequency to estimate the output distribution. Since all the datasets we use are open-ended generation tasks, different generated answers could have the exactly same meaning. For example, to answer the question *'When did the world's population reach 7 billion?'*, the LLM may generate several different answers such as *'December 2017'* and *'The world's population reached 7 billion in December 2017'*, which are essentially the same meaning. Regarding these two answers as distinct answers can lead to an overestimation of the entropy of output distribution. Previous work (Kuhn et al., 2022; Lin et al., 2023) uses a natural language inference model to cluster different generated sequences with the same semantic meanings into one group for better output distribution estimation. We empirically find that LLMs can achieve better clustering performance. Therefore, we prompt the LLM to cluster output answers into different groups for output distribution estimation on question-answering datasets. More details about this process can be found in Appendix A.6.

For all the experiments, we introduce the following baselines: Semantic Uncertainty (Kuhn et al., 2022) (denoted as SE) directly computes the entropy of the output distribution as the estimated (total) uncertainty (named *semantic entropy* in their paper). Tian et al. (2023) first queries the LLM for the answer and then queries the LLM again for the confidence of the correctness of the answer. We denote this method as ASK4CONF. We also slightly modify the prompt for the ambiguity detection task to query LLM for the confidence of the ambiguity of the input (denoted as ASK4CONF-D). The DEEP ENSEMBLES method (denoted as ENSEMBLES* for brevity) is implemented by ensembling the output distributions of multiple different in-context example sets (we use 5 different sets). We add * here since this method is different from standard DEEP ENSEMBLES and does not directly optimize Eq. 1. We provide more details of the prompts used in the experiments in Appendix A.4.

4.2. Quantifying Total Uncertainty

Correctly quantifying the total uncertainty is the premise of correctly decomposing the uncertainty. If the estimated total uncertainty is inaccurate, so will the estimated aleatoric and epistemic uncertainty. A reliable total uncertainty measure should have a close correspondence to the model's prediction accuracy. For model predictions whose total uncertainty is high, the chances that the predictions are incorrect should also be high. Therefore, we will evaluate the total uncertainty quantification using the mistake detection task, following the previous work (Kuhn et al., 2022; Lin et al., 2023).

Evaluation Settings We evaluate the total uncertainty on the `Natural Question` (`NQ`) dataset (Kwiatkowski et al., 2019) and `GSM8K` (Cobbe et al., 2021). For each

Table 1. Uncertainty quantification for mistake detection. Entropy (✔) refers to the average total uncertainty of questions with correct answers, while Entropy (✘) refers to the average total uncertainty of question with wrong answers.

Method	AUROC	F1 Score	Entropy (✔)	Entropy (✘)
Natural Question				
SEMANTIC ENTROPY	63.8	77.9	0.29	0.56
ASK4CONF	70.4	83.9	-	-
ENSEMBLES*	69.7	79.7	0.46	0.88
OURS	72.3	80.2	0.58	1.18
GSM8K				
SEMANTIC ENTROPY	88.2	92.4	0.32	1.46
ASK4CONF	58.1	92.3	-	-
ENSEMBLES*	88.3	94.6	0.57	1.94
OURS	89.7	94.7	0.42	1.82

dataset, we randomly sample 200 examples from the validation set for evaluation. The total uncertainty on each example is used to predict whether the model's answer is correct. We report the area under the receiver operator characteristic curve (AUROC) as well as the best F1 score when using the total uncertainty to predict the correctness of the model answer. We use 5-shot in-context examples on the `NQ` dataset and 2-shot on the `GSM8K` dataset with chain-of-thoughts. We prompt the LLM to rephrase the input question to generate the clarification set. The detailed prompts are listed in Appendix A.4.

Results The experiment results are shown in Table 1, which confirms that the total uncertainty measured by the proposed approach is reliable. Specifically, we highlight the following observations. First, our method achieves comparable uncertainty quantification performance compared to the baselines, achieving a similar AUROC and F1 score. Second, as the proposed method shares a symmetry form with the DEEP ENSEMBLES method, one would expect the total uncertainty quantification of the two should be similar. The above experimental results verify that the quantification results of these two methods are very close. Third, although ASK4CONF performs well on factual QA tasks, it provides a poor uncertainty estimation for the complex reasoning task (`GSM8K`), while our method can still provide good mistake detection performance.

4.3. Uncertainty Decomposition

Now we can proceed to evaluate whether the decomposed uncertainty is reliable. As discussed, one of the main causes of aleatoric uncertainty is the ambiguity of the input. Therefore, we will test how well the measured aleatoric uncertainty is predictive of whether an input is ambiguous. In particular, we focus on two input components, the instruction and the question, and separately predict the ambiguity within each component using the respective aleatoric uncertainty (see Section 3.5).

Datasets For ambiguity detection of the question, we select the `AmbigQA` dataset (Min et al., 2020), which has annotations on the ambiguity of questions. The questions in `AmbigQA` are extracted from the `NQ` dataset (Kwiatkowski et al., 2019). For ambiguity detection of the instruction, since there is no existing dataset, we create a dataset, `AmbigInst`, where we generate ambiguous instructions, their disambiguation, and the input-output pairs using ChatGPT. Each instruction is paired with around 15 questions. Since the focus of `AmbigInst` is to detect ambiguous instructions, we do not introduce ambiguity to the paired questions. More details about `AmbigInst` can be found in Appendix B. We use the full `AmbigInst` dataset and randomly sample 200 examples from the validation set of `AmbigQA` for evaluation.

Evaluation Settings We use 5-shot in-context examples on the `AmbigQA` dataset similar to the experiment on the `NQ` dataset. Since the questions in `AmbigInst` are relatively easy and straightforward, we directly prompt LLMs in a zero-shot setting. For ambiguous question detection, we perform clarifications on the input question only. We evaluate two clarification LLMs on the `AmbigQA` dataset, including the `Llama-3-8B-Instruct` fine-tuned on the training set of `AmbigQA` and `gpt-4`. we retrieve the most similar 16 questions as in-context examples when prompting the `gpt-4` to generate clarifications for a particular input question. The similarity between two questions is measured by the cosine similarity of their sentence embeddings from SENTENCE-BERT[1] (Reimers & Gurevych, 2019). For the `AmbigInst` dataset, we directly prompt `gpt-3.5-turbo-0613` to generate instruction clarifications (See Appendix A.4 for more details). We also include the performance of our method when using ground-truth disambiguation from the two datasets for reference (denoted as OURS*).

The baselines are similar to the methods in the mistake detection task. The main difference is we use the quantified uncertainty to predict whether the input contains ambiguity. The total uncertainty for SE is used for ambiguity prediction in this task. Also, we test both the aleatoric uncertainty and total uncertainty quantified by the DEEP ENSEMBLES method, denoted by ENSEMBLES*(aleatoric) and ENSEMBLES* (total) respectively. For our method, we use the aleatoric uncertainty for ambiguity prediction. DEEP ENSEMBLES is not included on the `AmbigInst` dataset since we do not include in-context examples on that dataset. We also incorporate results with additional methods from previous work (Si et al., 2022; Cole et al., 2023) in Appendix A.1.

[1]We use the pre-trained sentence transformer model https://huggingface.co/sentence-transformers/all-mpnet-base-v2.

Table 2. Uncertainty quantification for ambiguity d
AU (✔) refers to the average aleatoric uncertainty of
questions, while Avg. AU (✘) refers to the average a
tainty of ambiguous questions.

Method	AUROC	F1 Score	Avg. AU (✔)
AmbigQA			
SEMANTIC ENTROPY	54.9	46.8	0.24
ASK4CONF-D	55.0	64.3	-
ENSEMBLES* (aleatoric)	53.6	53.0	0.13
ENSEMBLES* (total)	55.4	55.0	0.50
OURS (GPT)	71.7	70.1	0.28
OURS (LLaMA)	67.1	71.8	0.55
OURS*	89.8	85.6	0.53
AmbigInst			
SEMANTIC ENTROPY	66.0	53.7	0.07
ASK4CONF-D	57.9	75.4	-
OURS (GPT)	81.3	77.9	0.10
OURS*	96.7	92.6	0.10

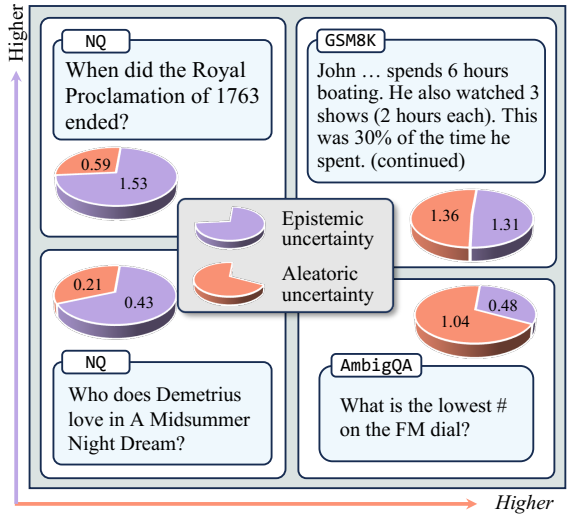

Figure 3. The uncertainty quantification examples using the proposed method. The instances are selected from existing datasets including `Natural Question (NQ)` (Kwiatkowski et al., 2019), `AmbigQA` (Min et al., 2020), and `GSM8K` (Cobbe et al., 2021).

Results The experiment results are shown in Table 2. We emphasize two observations. First, our method achieves the best ambiguity detection performance and significantly outperforms the baselines. Note that all the baselines, except for ENSEMBLES* (aleatoric), use the total uncertainty for ambiguity detection, and thus could not disentangle epistemic uncertainty from the aleatoric uncertainty. Therefore, these results verify the importance of uncertainty decomposition. Second, even a small model can also be efficiently adapted for the clarification generation task. When using the fine-tuned LLaMA model, Our method can still outperform baselines significantly. This adaptation process was remarkably efficient, requiring less than 10 minutes on 4×80G H100 GPUs. Third, the DEEP ENSEMBLES* (aleatoric) method is not effective in the black-box LLM setting. As we have discussed in Section 3.3, simply varying the in-context examples cannot accurately estimate the parameter posterior distribution, while the proposed framework is specially designed for the black-box LLMs.

Another observation is that ambiguity detection performance varies across different datasets. On the `AmbigQA` dataset (Min et al., 2020), the ambiguities are more implicit and hard to find by the clarification models, which makes the detection performance relatively low (although still higher than baselines significantly). Min et al. (2020) also note that the ambiguity in the dataset is "sometimes subtle" and "many (ambiguities) are only apparent after examining one or more Wikipedia pages". In comparison, on the `AmbigInst` dataset where we design ambiguities to be very explicit (see Appendix B for more examples), the clarification model can generate effective clarifications for most cases, leading to a good detection performance. Finally, the performance of our method can be further improved when using with the ground-truth disambiguation from the two datasets as the input clarifications, demonstrating that the clarification model is still worth exploring.

Qualitative Results We further present a visual representation of various uncertainty quantification results in Figure 3. These examples have been grouped according to the levels of aleatoric and epistemic uncertainty the LLM exhibits. Our uncertainty quantification framework enables a clear understanding of the sources of uncertainty in each example. For instance, consider the question "`What is the lowest # on the FM dial`" from the `AmbigQA` dataset. This question lacks specificity regarding the country and region, leading to ambiguity in the input. Our uncertainty quantification illustrates that the predominant source of uncertainty in the model's prediction stems from aleatoric uncertainty in this case. In contrast, despite a clear question, the model struggles to answer a query about the "Royal Proclamation" (as shown in the upper left example), resulting in a high level of epistemic uncertainty. Interestingly, we have identified a few examples within the `GSM8K` dataset where the uncertainty in the LLM prediction is attributed to data-related factors. For instance, the upper right example in the figure raises ambiguity about whether the word "`this`" refers solely to watching shows or encompasses both activities (the ground-truth annotation uses the second interpretation). More details about these examples can be found in Appendix A.5.

4.4. Monotonicity Check

To further evaluate the reliability of our aleatoric uncertainty measure, particularly the clarification module, we perform a monotonicity check experiment. Ideally, the clarified input should contribute to a much lower aleatoric uncertainty than the original ambiguous input. To test this, we perform two

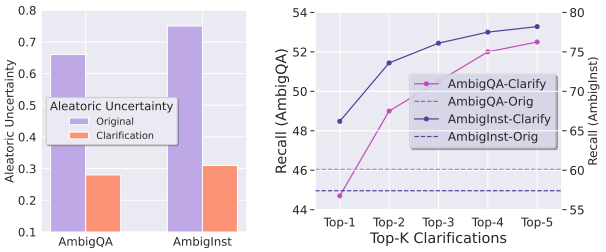

Figure 4. (Left) Average aleatoric uncertainty of the ambiguous inputs and their clarifications. (Right) Performance improvement via Soliciting clarifications. `AmbigQA-Orig` and `AmbigInst-Orig` refer to the recall of correct answers when directly answering the original input. `AmbigQA-Clarify` and `AmbigInst-Clarify` refer to the recall of correct answers using different number of input clarifications.

rounds of aleatoric uncertainty measuring. In the first round, we measure the aleatoric uncertainty by clarifying the original input segments (question or instruction). In the second round, we measure the aleatoric uncertainty of the clarified inputs obtained in the first round. Our goal is to check whether the aleatoric uncertainty measured in the second round is much smaller than that in the first round. This experiment is performed on the `AmbigQA` and `AmbigInst` datasets. In both rounds, we use the same clarification prompt to generate the clarifications.

Figure 4(a) visualizes the change in uncertainty on both datasets. As can be observed, the aleatoric uncertainty drops significantly after the input is clarified, which verifies the effectiveness of the clarification network.

4.5. Recall of Correct Answers

As discussed in Section 3.6, our framework can be used to improve the performance in the presence of ambiguous input by asking users to choose from a set of clarified versions of the input. To make this happen, our methods must be able to cover a good proportion of the possible answers resulting from different clarifications of a given ambiguous input. Also, the number of required clarifications should be smaller, as the users might not want to select the responses from a large set of choices.

To test this, we use the ambiguous questions and instructions from `AmbigQA` and `AmbigInst` respectively. For each input, we collect all the possible labeled answers from the ground-truth annotations. Then we select one answer as the target answer that the user is asking for. In our pipeline, the LLM will generate multiple answers given the generated clarifications. Therefore, we inspect how well these generated answers cover the target answer given different numbers of clarifications. We separately compute the recall of the target answer with the different numbers of clarifications. As a baseline, we introduce a vanilla version, where we directly query the LLM with ambiguous input without any clarification.

The results are illustrated in Figure 4(b). We can consistently observe an increase of recall given more clarifications. Similar to the ambiguity detection performance, the recall improvement on the `AmbigInst` dataset is more significant compared to the `AmbigQA` dataset, which is due to the subtlety of the `AmbigQA` dataset as discussed. Nevertheless, the proposed clarification framework is able to significantly improve the answer recall over the vanilla version without the clarification.

5. Conclusion

In this paper, we focus on the uncertainty quantification of LLMs and propose a new framework for decomposing the uncertainty. With a symmetric structure of the BNN methods, our framework leverages input clarifications for uncertainty quantification, which is more suitable for black-box LLMs. experimental results affirm that our proposed method not only yields reliable uncertainty quantification but also effectively decompose the total uncertainty into aleatoric and epistemic uncertainty. In the future, we will further explore how to build a more effective clarification module to boost the effectiveness of our method.

Acknowledgment

The work of Bairu Hou, Yujian Liu, and Shiyu Chang was partially supported by National Science Foundation (NSF) Grant IIS-2207052, NSF Grant IIS-2302730, and CAHSI-Google Research Award.

Impact Statement

In this paper, our primary objective is to develop an innovative uncertainty quantification framework, which aims to empower users in pinpointing the sources of uncertainty in LLMs accurately. The principal application scenario for our approach is to improve the trustworthiness and reliability of LLMs. Consequently, the likelihood of unintended usage or potential risks arising from our proposed method is considerably reduced. We also assess both the evaluations and datasets to ensure they are devoid of any harmful content or adverse impacts.

Nonetheless, it's imperative to note that our method relies on the well-calibrated nature of LLMs when applied to factoid QA and mathematical reasoning tasks. A well-calibrated LLM helps improve the reliability of uncertainty quantification. There exists a possibility that LLMs may not exhibit the same level of calibration on particular downstream tasks, and we are committed to continually refining the calibration of LLMs to bolster the reliability and trustworthiness of

machine learning systems.

References

Bhatt, U., Antorán, J., Zhang, Y., Liao, Q. V., Sattigeri, P., Fogliato, R., Melançon, G., Krishnan, R., Stanley, J., Tickoo, O., 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*, pp. 401–413, 2021.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural network. In *International conference on machine learning*, 2015.

Chen, J. and Mueller, J. Quantifying uncertainty in answers from any language model via intrinsic and extrinsic confidence assessment. *arXiv preprint arXiv:2308.16175*, 2023.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Cole, J. R., Zhang, M. J., Gillick, D., Eisenschlos, J. M., Dhingra, B., and Eisenstein, J. Selectively answering ambiguous questions. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

Desai, S. and Durrett, G. Calibration of pre-trained transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

Duan, J., Cheng, H., Wang, S., Wang, C., Zavalny, A., Xu, R., Kailkhura, B., and Xu, K. Shifting attention to relevance: Towards the uncertainty estimation of large language models. *arXiv*, 2023.

Fort, S., Hu, H., and Lakshminarayanan, B. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059. PMLR, 2016.

Gal, Y. et al. Uncertainty in deep learning, 2016.

Graves, A. Practical variational inference for neural networks. *Advances in neural information processing systems*, 24, 2011.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *International conference on machine learning*, pp. 1321–1330. PMLR, 2017.

Guo, M., Zhang, M., Reddy, S., and Alikhani, M. Abgcoqa: Clarifying ambiguity in conversational question answering. In *3rd Conference on Automated Knowledge Base Construction*, 2021.

Hasenclever, L., Webb, S., Lienart, T., Vollmer, S., Lakshminarayanan, B., Blundell, C., and Teh, Y. W. Distributed bayesian learning with stochastic natural gradient expectation propagation and the posterior server. *Journal of Machine Learning Research*, 18(106):1–37, 2017.

He, B., Lakshminarayanan, B., and Teh, Y. W. Bayesian deep ensembles via the neural tangent kernel. *Advances in neural information processing systems*, 33:1010–1022, 2020.

Hendrycks, D. and Gimpel, K. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

Hernández-Lobato, J. M. and Adams, R. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International conference on machine learning*, pp. 1861–1869. PMLR, 2015.

Honovich, O., Shaham, U., Bowman, S. R., and Levy, O. Instruction induction: From few examples to natural language task descriptions. *arXiv preprint arXiv:2205.10782*, 2022.

Huang, J., Gu, S. S., Hou, L., Wu, Y., Wang, X., Yu, H., and Han, J. Large language models can self-improve. *arXiv preprint arXiv:2210.11610*, 2022.

Huang, Y., Song, J., Wang, Z., Chen, H., and Ma, L. Look before you leap: An exploratory study of uncertainty measurement for large language models. *arXiv preprint arXiv:2307.10236*, 2023.

Hüllermeier, E. and Waegeman, W. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110:457–506, 2021.

Jiang, M., Ruan, Y., Huang, S., Liao, S., Pitis, S., Grosse, R. B., and Ba, J. Calibrating language models via augmented prompt ensembles. 2023.

Jiang, Z., Araki, J., Ding, H., and Neubig, G. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 2021.

Kadavath, S., Conerly, T., Askell, A., Henighan, T., Drain, D., Perez, E., Schiefer, N., Hatfield-Dodds, Z., DasSarma, N., Tran-Johnson, E., et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Koller, A., Regneri, M., and Thater, S. Regular tree grammars as a formalism for scope underspecification. In *Proceedings of ACL-08: HLT*, pp. 218–226, 2008.

Kristiadi, A., Hein, M., and Hennig, P. Learnable uncertainty under laplace approximations. In *Uncertainty in Artificial Intelligence*, pp. 344–353. PMLR, 2021.

Kuhn, L., Gal, Y., and Farquhar, S. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The Eleventh International Conference on Learning Representations*, 2022.

Kuhn, L., Gal, Y., and Farquhar, S. Clam: Selective clarification for ambiguous questions with generative language models. 2023.

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019.

Lahlou, S., Jain, M., Nekoei, H., Butoi, V. I., Bertin, P., Rector-Brooks, J., Korablyov, M., and Bengio, Y. Deup: Direct epistemic uncertainty prediction. *Transactions on Machine Learning Research*, 2022.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

Li, Y., Hernández-Lobato, J. M., and Turner, R. E. Stochastic expectation propagation. *Advances in neural information processing systems*, 28, 2015.

Lin, S., Hilton, J., and Evans, O. Teaching models to express their uncertainty in words. *Transactions on Machine Learning Research*, 2022.

Lin, Z., Trivedi, S., and Sun, J. Generating with confidence: Uncertainty quantification for black-box large language models. *arXiv preprint arXiv:2305.19187*, 2023.

Liu, A., Wu, Z., Michael, J., Suhr, A., West, P., Koller, A., Swayamdipta, S., Smith, N. A., and Choi, Y. We're afraid language models aren't modeling ambiguity. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

Louizos, C. and Welling, M. Structured and efficient variational deep learning with matrix gaussian posteriors. In *International conference on machine learning*, pp. 1708–1716. PMLR, 2016.

Malinin, A. and Gales, M. Predictive uncertainty estimation via prior networks. *Advances in neural information processing systems*, 31, 2018.

Malinin, A. and Gales, M. Uncertainty estimation in autoregressive structured prediction. In *International Conference on Learning Representations*, 2020.

Malinin, A., Prokhorenkova, L., and Ustimenko, A. Uncertainty in gradient boosting via ensembles. In *International Conference on Learning Representations*, 2020.

Mielke, S. J., Szlam, A., Dinan, E., and Boureau, Y.-L. Reducing conversational agents' overconfidence through linguistic calibration. *Transactions of the Association for Computational Linguistics*, 2022.

Min, S., Michael, J., Hajishirzi, H., and Zettlemoyer, L. Ambigqa: Answering ambiguous open-domain questions. *arXiv preprint arXiv:2004.10645*, 2020.

Mobiny, A., Yuan, P., Moulik, S. K., Garg, N., Wu, C. C., and Van Nguyen, H. Dropconnect is effective in modeling uncertainty of bayesian deep networks. *Scientific reports*, 11:5458, 2021.

Neal, R. M. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

Ott, M., Auli, M., Grangier, D., and Ranzato, M. Analyzing uncertainty in neural machine translation. In *International Conference on Machine Learning*, pp. 3956–3965. PMLR, 2018.

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J., Lakshminarayanan, B., and Snoek, J. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems*, 32, 2019.

Park, S. and Kim, T. Pac neural prediction set learning to quantify the uncertainty of generative language models. *arXiv preprint arXiv:2307.09254*, 2023.

Reimers, N. and Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019.

Ren, A., Dixit, A., Bodrova, A., Singh, S., Tu, S., Brown, N., Xu, P., Takayama, L., Xia, F., Varley, J., et al. Robots that ask for help: Uncertainty alignment for large language model planners. In *2nd Workshop on Language and Robot Learning: Language as Grounding*, 2023.

Riquelme, C., Tucker, G., and Snoek, J. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. In *International Conference on Learning Representations*, 2018.

Shen, M., Bu, Y., Sattigeri, P., Ghosh, S., Das, S., and Wornell, G. Post-hoc uncertainty learning using a dirichlet meta-model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 9772–9781, 2023.

Si, C., Gan, Z., Yang, Z., Wang, S., Wang, J., Boyd-Graber, J. L., and Wang, L. Prompting gpt-3 to be reliable. In *The Eleventh International Conference on Learning Representations*, 2022.

Si, C., Gan, Z., Yang, Z., Wang, S., Wang, J., Boyd-Graber, J. L., and Wang, L. Prompting GPT-3 to be reliable. In *The Eleventh International Conference on Learning Representations*, 2023.

Tamkin, A., Handa, K., Shrestha, A., and Goodman, N. Task ambiguity in humans and language models. In *The Eleventh International Conference on Learning Representations*, 2022.

Teye, M., Azizpour, H., and Smith, K. Bayesian uncertainty estimation for batch normalized deep networks. In *International Conference on Machine Learning*, pp. 4907–4916. PMLR, 2018.

Tian, K., Mitchell, E., Zhou, A., Sharma, A., Rafailov, R., Yao, H., Finn, C., and Manning, C. D. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. *arXiv preprint arXiv:2305.14975*, 2023.

Wang, X., Wei, J., Schuurmans, D., Le, Q. V., Chi, E. H., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2022.

Xiao, Y., Liang, P. P., Bhatt, U., Neiswanger, W., Salakhutdinov, R., and Morency, L.-P. Uncertainty quantification with pre-trained language models: A large-scale empirical analysis. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 7273–7284, 2022.

Ye, X. and Durrett, G. Can explanations be useful for calibrating black box models?, 2022.

Zhou, K., Jurafsky, D., and Hashimoto, T. Navigating the grey area: Expressions of overconfidence and uncertainty in language models. *arXiv preprint arXiv:2302.13439*, 2023.

A. Additional Results and Implementation Details

A.1. Additional Results

In this section, we include additional methods that estimate the model's confidence on its answer to study whether the confidence score can be used for ambiguity detection. Specifically, Si et al. (2022) uses a self-consistency frequency to estimate the LLM confidence (denoted as SELFCONSISTENCY), where multiple answers are sampled from the LLM given the original question and the highest answer frequency are used as the confidence score. In addition, Cole et al. (2023) incorporate two features to estimates the LLM confidence: the answer repetition and the answer diversity. The answer repetition parallels the implementation of Si et al. (2022), and the answer diversity is estimated by counting the number of unique answers from multiple sampled answers. We denote the two methods as SAMPLE REPETITION and SAMPLE DIVERSITY. We include these 3 confidence scores for ambiguity detection.

We implement these methods as follows. For SELFCONSISTENCY, we use the default hyperparameters in the official implementation (temperature = 0.7, sample 10 answers from the model) to compute the confidence. When predict the input ambiguities, we use 1-confidence as the input, since a lower confidence implies either the model's answer is wrong or there are multiple valid answers (i.e., the input is ambiguous). For SAMPLEDIVERSITY, we use the default hyperparameters in the official implementation (temperature = 0.5, sample 10 answers from the model). We computed the number of unique answers, using this metric to gauge the ambiguity of the input question. For SAMPLEREPETITION, this method parallels SELFCONSISTENCY, where answers are generated using greedy decoding and then re-sampled (temperature = 0.5, sample 10 answers) to assess the initial answer's confidence.The experiment results is as below:

Table 3. Additional results for ambiguity detection. Avg. AU (✔) refers to the average aleatoric uncertainty of unambiguous questions, while Avg. AU (✘) refers to the average aleatoric uncertainty of ambiguous questions.

Method	AUROC	F1 Score	Avg. AU (✔)	Avg. AU (✘)
AmbigQA				
SEMANTIC ENTROPY	54.9	46.8	0.24	0.47
ASK4CONF-D	55.0	64.3	-	-
ENSEMBLES* (aleatoric)	53.6	53.0	0.13	0.13
ENSEMBLES* (total)	55.4	55.0	0.50	0.41
SELFCONSISTENCY	56.0	62.5	-	-
SAMPLE REPETITION	58.6	69.0	-	-
SAMPLE DIVERSITY	57.6	66.7	-	-
OURS (GPT)	71.7	70.1	0.28	0.67
OURS (LLaMA)	67.1	71.8	0.55	0.91
OURS*	89.8	85.6	0.53	1.52

We visualize the results in Table 3. These results underscore that model confidence alone is insufficient for accurately identifying ambiguous inputs, as evidenced by the AUROC scores of all three methods being under 60. We will also include these findings and citations in the final version of the paper.

A.2. Supervised Fine-tuning for Clarification Generation

We fine-tuning the `Llama-3-8B-Instruction` on the full training set of `AmbigQA` dataset on 4×NVIDIA H100 80GB HBM3 GPU. We organize the training data using the template in Figure 5. We use `PyTorch Lightning`, `DeepSpeed` Stage 1, and `flash-attention 2` to train the model. We train the model with batch size 16, learning rate 2e-5, and cosine learning rate scheduler for 5 epochs. The loss is only computed on the output tokens. We evaluate the model on the validation set and take the model that achieves lowest validation loss (epoch = 2) for testing.

A.3. Implementation details for baselines

Mistake detection For the mistake detection task, we strictly follow the experiment settings from Kuhn et al. (2022) and Lin et al. (2023). For each example, we estimate the output distribution and take the answer with the highest frequency as the final answer. Then we use the method (and the prompt) from Lin et al. (2023) to determine whether the answer is correct by prompting ChatGPT. Based on the total uncertainty and correctness of the answer, we compute the AUROC and conduct a grid search to find the best threshold for the F1 score, where the correct answers are regarded as positive examples.

For the implementation of ASK4CONF(Tian et al., 2023) in the mistake detection task, we use the "Verb. 2S top-1" method (and the corresponding prompts) to estimate the confidence of the language model. Rather than asking the LLM to directly generate an answer, we sample multiple answers and take the most frequent one as the answer. After that, we prompt the LLM for the confidence of the most frequent answer. The prompt we use is:

```
<|begin_of_text|><|start_header_id|>user<|end_header_id|>

In this task, you will receive a question that may contain ambiguities. First analyze the
following aspects to find if there is any ambiguities according to the real-world facts:
- entities, objects, or events has multiple references or interpretations
- Unclear timestamps
- Unclear locations
- Unclear answer types (e.g., "When" refers to "which year or what date", and "Who" refers to "
which person or which team")

If there is any ambiguities, you need to remove ambiguities by adding some clarifications to
the question. Each clarification is an additional condition or explanations to the concept in
the question that resolve its ambiguity.
    - You are only allowed to add conditions or explanations, and you cannot change the
original intent or semantics of the question.
    - The conditions and explanations must be ground to real-word facts.

If there is no ambiguities, you only need to output the original question as it is.<|eot_id|><|
start_header_id|>assistant<|end_header_id|>

Sure. Please provide me with the question so that I can identify whether it is ambiguous and
clarify it.<|eot_id|><|start_header_id|>user<|end_header_id|>

Original Question: {original_question}<|eot_id|><|start_header_id|>assistant<|end_header_id|>

Question after adding condition: {ground_truth_clarification}<|eot_id|>
```

Figure 5. The prompt template for the fine-tuning of `Llama-3-8B-Instruction`. The {`original_question`} and {`ground_truth_clarification`} are two placeholders that will be filled with the original question (either ambiguous or not) and the ground-truth clarifications.

Ambiguity detection For the mistake detection task, we use the total uncertainty for SEMANTIC UNCERTAINTY (Kuhn et al., 2022), aleatoric uncertainty from BNN*, and the confidence score of the ambiguity from ASK4CONF (Tian et al., 2023) to predict whether the input is ambiguous or not. We slightly modify the prompt of ASK4CONF as follows:

A.4. Prompts for Our Clarification Model

We list the prompts we used for clarification generation on each dataset as follows:

- Input clarification prompt on `Natural Question` and `GSM8K` is shown in Figure 8.
- Input clarification prompt on `AmbigQA` is shown in Figure 9.
- Input clarification prompt on `AmbigInst` is shown in Figure 10.

A.5. Details of the Qualitative Results

Due to space limit, we truncate the example from `GSM8K` visualized in Figure 3. The whole question is: "John decides to do several activities while out on vacation. He spends 6 hours boating and half that time swimming. He also watched 3 different shows which were 2 hours each. This was 30% of the time he spent. He spent 40% of his time sightseeing. How much time did he spend sightseeing?".

When generating the clarifications, we directly prompt the LLM to paraphrase the question with the following prompt: "I am confused with the following question. Please paraphrase it so that it is easier to understand and solve." Then we sample 5 responses from the LLM as the clarifications for uncertainty quantification.

A.6. Prompt the LLM for Answer Extraction

As we have discussed in Section 4.1, different outputs generated by the LLM may have the same meaning in the free-form text generation setting. Unlike previous work (Kuhn et al., 2022; Lin et al., 2023) that map semantics-equivalent answers

```
Answer the following question.
Question: {The testing question}
Answer: {The most frequent answer}

Provide the probability that your answer is correct. Give ONLY the probability, no other words
or explanation.

For example:

Probability: <the probability between 0.0 and 1.0 that your solution is correct, without any
extra commentary whatsoever; just the probability!>
```

Figure 6. The prompt for mistake detection (ASK4CONF).

```
Read the following question:
Question:
{question}
Provide the probability that this question is ambiguous due to factors such as ambiguous
entities, ambiguous event references, or ambiguity over the answer type. Give ONLY the
probability, no other words or explanation.

For example:

Probability: <the probability between 0.0 and 1.0 that the question is ambiguous (1.0 means the
 question is absolutely ambiguous), without any extra commentary whatsoever; just the
probability!>
```

Figure 7. The prompt for ambiguity detection (ASK4CONF-D).

into a unique set using the NLI models, we empirically find that the LLMs provide better performance on this task. The prompt we use is in Figure 11.

After we extract and cluster the semantically equivalent answers, we further post-process the answers as follows. First, the clarifications generated by the clarification LLM may be invalid and have no answers. In such cases, the LLM may refuse to respond to the question and reply with phrases like "I'm sorry, but I couldn't find any information about the question" or other similar replies. The answer extraction prompt in Figure 11 maps these answers to a special answer, "Unknown." To ensure these answers are mapped to "Unknown," we adopt a keyword-matching approach that defines a set of key phrases signaling a refusal to respond, such as "I'm sorry," "cannot be determined," and "invalid question." Answers containing these key phrases are mapped to "Unknown". Second, if all answers to a particular clarification are mapped to "Unknown", we regard this clarification as invalid and directly drop it when ensembling the outputs for uncertainty quantification. Otherwise, the appearance of the answer "Unknown" indicates the model's insufficient knowledge regarding the question, contributing to the epistemic uncertainty. Therefore, when computing the frequency to estimate the output distribution, we count the occurrences of each unique answer, excluding the special "Unknown". Then, we normalize these counts by dividing each by the total number of answers. For every appearance of the special "Unknown", we increase the normalized frequencies of all other answers by $\frac{1}{N}$, where N is the number of unique answers excluding the special answer. This adjustment ensures the special answer's impact is evenly distributed across the other answers and increases the epistemic uncertainty.

B. `AmbigInst` Dataset

B.1. Dataset Creation

We generate ambiguous instructions following the pipeline of SELF-INSTRUCTION (Wang et al., 2022). Specifically, we first query CHATGPT with several manually designed ambiguous task descriptions as in-context examples. For better verification of the ambiguity, we also prompt CHATGPT to output the cause of the ambiguity. Among the ambiguous

```
In this task, you will receive a single question, and your goal is to generate multiple
versions of it that convey the same meaning as the original. Please format your responses as
follows:
Rephrase 1: [Your rephrased question]
Rephrase 2: [Another rephrased question]
Rephrase 3: [Yet another rephrased question]
....
Ensure that each rephrased question is distinct from the others."

Here are two examples:
(examples skipped)
```

Figure 8. The prompt for question rephrase on the `Natural Question` dataset

descriptions generated by CHATGPT, we manually filter out those that have an open-ended output space such as `Write a report on the new marketing campaign`. The final dataset contains 15 ambiguous task descriptions. After that, we query CHATGPT again to generate ground-truth clarifications based on the cause of ambiguities generated in the first query.

Given the collected ambiguous task descriptions and their clarifications, we then prompt the model to generate input-output pairs for each task. Specifically, 15 inputs are generated for each task, and each input is further paired with different output answers depending on the ground-truth clarifications. We additionally add a post-processing step where we filter out the inputs that have exactly the same answer given different clarifications. The final ambiguous instructions consist of 15 tasks with 214 input questions in total.

We take 10 tasks from the `Instruction induction` dataset (Honovich et al., 2022) as the unambiguous tasks, including `letters_list`, `first_word_letter`, `second_word_letter`, `orthography_starts_with`, `larger_animal`, `singular_to_plural`, `diff`, `num_to_verbal`, `antonyms`, and `sum`.

We manually add some clarifications to the 10 instructions to remove potential ambiguities. For example, given the original instruction "`Break the input word into letters, separated by spaces`", we clarify it with "`Write the inputted word with a space between each letter`", since "separated by spaces" might cause ambiguities of how many spaces should be added between two letters. Each task is also paired with 15 input-output pairs. Overall, the `AmbigInst` dataset contains 25 tasks and 364 different inputs.

B.2. Dataset Examples

We list several examples from the synthetic dataset with ambiguous instructions.

▷ 1. Rearrange the objects on the table in ascending order.

Input: The following table lists the objects on my desk:

Name	Size	Weight	Color	Date of Manufacture	Price
Pen	14cm	0.02kg	blue	01/15/2022	$1.50
Book	23cm	0.5kg	red	08/10/2020	$15.00
Laptop	38cm	1.8kg	silver	05/04/2021	$1200.00

▷ 2. Calculate the average of the numbers in the given list, rounding to the nearest whole number.

Input: 23.5, 47.2, 30.1, 16.6

▷ 3. Determine the length of a sentence.

```
In what follows, you will be given some questions that might be ambiguous. These ambiguities
can arise from various factors, including but not limited to:

1. Ambiguous references to entities in the question.
2. Multiple properties of objects/entities in the question leading to different interpretations
.
3. Ambiguities due to unclear timestamps.
4. Ambiguities stemming from unclear locations.
5. Multiple valid answer types based on the question.

For each question, you are to provide at least two distinct rephrasings that resolve these
ambiguities. By "rephrasing," we mean you should reformulate the question to be clear and
direct, eliminating any possible ambiguity without altering the original intent of the question
. You should not seek further information or produce a binary (yes-no) question as a result of
the clarification. Instead, you must create a direct question (wh-question) that aims to obtain
 a specific answer.


Please format your responses as follows (with at least two rephrasings per question):
Clarifications:
1. [First rephrased question]
2. [Second rephrased question]
3. [Third rephrased question]
...

If the original question is already clear and unambiguous, you should indicate this by stating,
 "No clarification needed."

(In-context examples)
```

Figure 9. The prompt for question disambiguation on the `AmbigQA` dataset.

Input: The quick brown fox jumps over the lazy dog.

▷ 4. Sort the names alphabetically.

Input: Courtney Cox, Jennifer Aniston, Lisa Kudrow, Matthew Perry.

▷ 5. Identify the subject in the sentence.

Input: The CEO of the company gave a speech about the future of technology.

▷ 6. Count the number of objects in the given list of objects.

Input: Forks, Spoons, Knives, Plates, Cups, Spoons, Forks, Spoons, Cups.

▷ 7. Rank the football players based on their performance.

Input: The following table lists the statistics of football players:

Name	Goal Scored	Assists
Lionel Messi	30	12
Cristiano Ronaldo	25	10
Robert Lewandowski	35	5

```
**Objective**
Analyze the given task description for ambiguities based on the description itself and the
provided input question. If the task description is ambiguous, your task is to clarify it by
interpreting the ambiguous concepts, specifying necessary conditions, or using other methods.
Provide all possible disambiguations.

**Important Rules**
1. Perform detailed analyses before concluding whether the task description is clear or
ambiguous.
2. Output disambiguations in the specified format.
3. Some seemingly unambiguous task descriptions are actually ambiguous given that particular
input. So, do not forget to leverage the input to analyze whether the task description is
underspecified.

**Output Format**
Your output should follow this format:
Analyses:
[Think step-by-step to reason on the clarity of the task description. After that, output your
judgement on whether the task description is ambiguous or not]

Disambiguations:
1. [Disambiguated task description 1.]
2. [Disambiguated task description 2.]
3. [Disambiguated task description 3.]
...

If the task description is clear and unambiguous, simply output:
Disambiguations:
1. No clarification needed.
```

Figure 10. The prompt for instruction disambiguation on the `AmbigInst` dataset.

▷ 8. Sort the data in alphabetical order.

Input: Dog, Cat, Bird, Fish, Aardvark.

▷ 9. Identify the largest city in the set. Input: The following table lists the cities in the set:

Name	Population	Land Area
Paris	2.1 million	105.4 km
Berlin	3.6 million	891.8 km
Madrid	3.3 million	604.3 km

▷ 10. Organize the files by date.

Input: Files to be organized:

Filename	Creation Date	Last Modified Date
conference-recording.avi	11/10/2020	11/12/2020
birthday-video.mp4	05/05/2021	05/06/2021
budget.xlsx	12/31/2022	01/10/2023

▷ 11. Find the middle value in a list of numbers.

```
**Task: Answer Extraction from Sentences**

In this task, you will receive both a question and multiple sentences. Each sentence contains
an answer to the question. Your primary goal is to extract a concise answer, which can be a
single word or a short phrase, from each sentence. Again, ensure you only extract a short
answer! If a short answer cannot be directly extracted, then summarize the whole sentence into
a single word or a short phrase.

Additionally, while extracting answers, your secondary goal is to create an "answer set" that
contains all distinct answers from previous questions. If the extracted answer has not appeared
 in the answer set, add it to the answer set.

**Important Rules**
1. If there is an answer in the answer set that is semantically equivalent to the extracted
answer, use the answer from the answer set as the result. Do not introduce a new, slightly
different answer. For example, if the answer set already contains "the matrix (1999)," and you
extract an answer from a sentence like "The popular movie in 1999... is the matrix," your
extraction should be "the matrix (1999)" rather than "the matrix."

2. Separate different answers in the answer set using "|".

3. Also, extract the answer as "Unknown" for the following cases:
    - The sentence claims that there is no answer to the question
    - The sentence claims it lacks sufficient information to answer the question
    - The sentence claims it depends on various factors and the answer cannot be determined


**Output Format**

Your output format should follow this pattern (N is the number of sentences):

Answer set at the beginning: [ ]
Extraction 1/N: [extraction from 1st sentence]
Updated answer set: [ ]
Extraction 2/N: [extraction from 2nd sentence]
Updated answer set: [ ]
Extraction 3/N: [extraction from 3rd sentence]
Updated answer set: [ ]
...
Final answer set: [ ]

**Example**
(examples skipped)
```

Figure 11. The prompt for answer extraction using LLMs for the `Natural Question` and `AmbigQA` datasets.

Input: 12, 20, 35, 46, 52, 66, 74, 81

▷ 12. Determine the square root of a number.

Input: 81

▷ 13. Find the capital of a country.

Input: South Africa

▷ 14. Classify a movie based on its rating.

Input: The movie “Toy Story 4” has an MPAA rating of G, an IMDb rating of 7.8, and a Rotten Tomatoes rating of 97%.

▷ 15. Select the longest sentence from the following choices, and output the sentence index.

Input: The following sentences are listed:

1. To be, or not to be, that is the question.
2. Whether 'tis nobler in the mind to suffer the slings and arrows of outrageous fortune.
3. Or to take arms against a sea of troubles and by opposing end them.