Do not Abstain! Identify and Solve the Uncertainty

Anonymous ACL submission

Abstract

Despite the widespread application of Large Language Models (LLMs) across various domains, they frequently exhibit overconfidence when encountering uncertain scenarios, yet existing solutions primarily rely on evasive responses (e.g., "I don't know") overlooks the opportunity of identifying and addressing the uncertainty to generate more satisfactory responses. To systematically investigate and improve LLMs' ability of recognizing and addressing the source of uncertainty, we introduce **ConfuseBench**, a benchmark mainly focus on three types of uncertainty: document scarcity, limited capability, and query ambiguity. Experiments with ConfuseBench reveal that current LLMs struggle to accurately identify the root cause of uncertainty and solve it. They prefer to attribute uncertainty to query ambiguity while overlooking capability limitations, especially for those weaker models. To tackle this challenge, we first generate context-aware inquiries that highlight the confusing aspect of the original query. Then we judge the source of uncertainty based on the uniqueness of the inquiry's answer. Further we use an on-policy training method, InteractDPO to generate better inquiries. Experimental results demonstrate the efficacy of our approach.

1 Introduction

Large Language Models (LLMs) (Brown, 2020; Li et al., 2023; Wu et al., 2023) have demonstrated remarkable capabilities in a variety of tasks, including text generation, question answering (Ouyang et al., 2022; Wei et al., 2023), code generation (Gu, 2023), information retrieval (Dai et al., 2024) and tool use (Qin et al., 2023). However, LLMs tend to exhibit a significant degree of overconfidence (Xiong et al., 2024; Li et al., 2024b) when faced with question they are not aware of.

To mitigate this issue, existing researches primarily adopt conservative strategies: response with "I don't know" when identifying potential uncertainties (Amayuelas et al., 2024; Deng et al., 2024; Li et al., 2024a; Madhusudhan et al., 2024). However, this strategy exhibits significant limitations. As shown in Table 1, for inherently unknowable questions (e.g., "weather of 2050."), models should consistently response with "I don't know". However, for those answerable queries, simply response with "I do not know" overlooks the opportunity of addressing the uncertainty, failing to generate more satisfactory responses. Specifically, when confidence levels are low (e.g., "quantum computing's impact on climate modeling"), the system should proactively identify uncertainty sources (insufficient document/reasoning capability gap/query ambiguity), then employ dynamic strategies such as retrieval (Lewis et al., 2020; Zhang et al., 2024a), CoT (Wei et al., 2023; Li et al., 2024c), or clarification (Qian et al., 2024; Yang et al., 2024a) to improve the response quality.

Query	Uncertainty	LLM Response	Expectation
unanswerable	low	I do not know	I do not know
unanswerable	high	hallucinate	I do not know
answerable	low	answer	answer
answerable	high	hallucinate	solve it

Table 1: Different behavior of LLM when faced with different query. "unanswerable" mean query can not be answered like "weather of 2050."

To investigate and improve LLMs' performance on identifying and solving the uncertainty, we introduce **ConfuseBench**, a benchmark that encompasses three distinct types of uncertainty: document scarcity, limited capacity, and query ambiguity. Document scarcity occurs when models lack essential factual information to answer a question, and additional documents could provide assistance (Lewis et al., 2020; Zhang et al., 2024a); Limited capacity indicates that the query is too complex for the model to resolve effectively, in such cases, a larger model or extended reasoning steps might be beneficial (Wei et al., 2023; Yao et al., 2024). Query ambiguity occur when the query itself is unclear, where multiple answers may suffice or the query may not be answerable at all, necessitating further clarification (Min et al., 2020; Qian et al., 2024; Zhang et al., 2024b). Through ConfuseBench, we surpass conventional evaluation paradigms that focus solely on answer accuracy or basic uncertainty detection. Instead, ConfuseBench rigorously assesses models' capacity to (1) diagnose the root causes of uncertainty and (2) actively mitigate such uncertainty to generate substantively improved responses.

Our experiments with ConfuseBench have revealed that current models including GPT-4o struggle to identify the sources of uncertainty, which leads to unsatisfying performance on this benchmark. Those models prefers to categorize questions as ambiguous and request the user for clarification. For example, when we provide the model a clear query "locate the best yoga class in New York" and a noise document about yoga classes in London, the model might regard the query as ambiguous and response with "Are you referring to London?". Furthermore, the models seldomly acknowledge failures caused by their own capability limitations, when confronted with uncertainty, models often attribute the issue to external factors rather than recognizing their own limitations.

To address this issue, we propose a two-step approach. Instead of directly identifying the source of uncertainty, we first focus on accurately locating the confusing parts of the problem and generating a follow-up inquiry. If the answer to inquiry is an objective fact, the retrieval system can effectively provide the required information. If multiple answers fit the inquiry appropriately, further clarification is necessary. Conversely, if the follow-up inquiry is logically incoherent or merely paraphrased repetitions of the original question, it means the model fails to effectively understand the query and CoT could be beneficial. Furthermore, to enhance the capability of generating effective follow-up inquiry, we propose the InteractDPO, a training paradigm that dynamically generates "chosen-rejected" sample pairs through real-time interaction with retrieval systems or users during training, thereby achieving on-policy optimization.

Overall, our key contributions include: 1) This paper introduces a new benchmark designed to measure LLMs' ability to identify different types

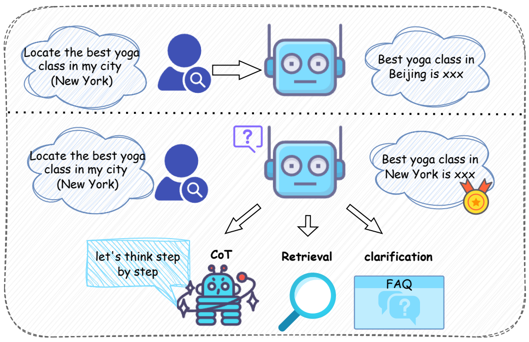

Figure 1: LLMs recognize different source of uncertainty and try to solve the uncertainty.

of uncertainty arising from various sources, including document scarcity, limited capability, and query ambiguity. 2) We demonstrate through experiments that current LLMs exhibit significant challenges in reliably differentiating between these sources of uncertainty. 3) We propose a novel method for identifying the source of uncertainty based on the uniqueness of the inquiry's answer, and we further enhance the inquiry generation process through InteractDPO.

2 Related Work

Recognizing the Uncertainty. Amayuelas et al. (2024); Yin et al. (2023) propose that models should learn to understand what they do not know instead of giving a hallucinated answer. Slobodkin et al. (2023); Amayuelas et al. (2024); Madhusudhan et al. (2024) further design various of prompts to instruct LLM express "I do not know" when encountering uncertainty. Yang et al. (2024b); Xiong et al. (2023) try to finetune the model to express how uncertain it is in verbal language and Deng et al. (2024) propose to train the model to give some explanation to the unanswerability. Deng et al. (2024); Zhang et al. (2024b) also categorize why a question is unknown, but they mainly focus on ill-defined input, ignoring lack of capacity and documents, they still fail to recognize and solve the source of uncertainty.

Solving the Uncertainty. For knowledge based uncertainty, Trivedi et al. (2022); Wang et al. (2024a) try to solve multi-hop queries by iterative reasoning and retrieving until the model feels confident enough to provide an answer, Jeong et al. (2024); Wang et al. (2024b); Zhuang et al. (2024) also iteratively call the model to generate retrieval query to solve the uncertainty. For ambiguity based uncertainty, Qian et al. (2024) construct Intention-in-Interaction (IN3) to evaluate the ability of asking clarification question, Wang et al. (2024c) prompts the LLM to adaptively ask clarification questions and Yang et al. (2024a) further propose to use prompt, entropy and logits to measure is clarification needed. However, these works are constrained to only one source of uncertainty, fail to consider the situation that the uncertainty may rise from other sources.

Uncertainty Decomposition. As uncertainty could be raised from different sources, recognizing the source is an important topic (Wang and Holmes, 2024; Geng et al., 2024; Huang et al., 2024), previous works typical classify uncertainty into data and model uncertainty. Hou et al. (2023) try to judge does the query needs to be clarified by observing how will the model perform when faced with different clarifications. Ling et al. (2024) use ensemble methods and use different in context example to simulate models to decompose the uncertainty. Yadkori et al. (2024) propose a new definition of model uncertainty and decompose uncertainty by distribution shift when some answers are provided to the LLM. But those methods simply classifies the uncertainty as data uncertainty and model uncertainty, fails to consider the real uncertainty types the LLM would met in application.

3 Benchmark Construction

Previous benchmarks have primarily focused on refusing to answer unknown queries (Amayuelas et al., 2024; Deng et al., 2024), or have merely considered iterative retrieval and clarification techniques (Wang et al., 2024b; Zhuang et al., 2024; Qian et al., 2024). This approach fails to recognize that models need to identify the source of uncertainty and implement corresponding measures to address it. To comprehensively enhance and quantitatively evaluate these capabilities in model designs, we introduce **ConfuseBench**, a benchmark that encompasses various sources of uncertainty. This benchmark aims to assess and inspire LLMs' abilities to recognize and resolve uncertainties effectively.

We evaluate three main scenarios in which LLMs are commonly employed: basic question answering, assistant interactions, and tool utilization. To assess the ability of recognizing and resolving uncertainty, we have collected various datasets and rewritten queries and associated documents to create the case holding certain uncertainty. For basic question answering, we incorporate HotpotQA(Yang et al., 2018) and AmbigQA (Min et al., 2020). In the assistant scenario, we consider ExpertQA (Malaviya et al., 2024) and TechQA (Castelli et al., 2019). We utilize ToolBench (Qin et al., 2023) for tool usage. It is worth noting that, to facilitate this evaluation, we employ GPT-4o to generate the tool calling chain for ToolBench, using the calling chain as the answer rather than the actual calling result.

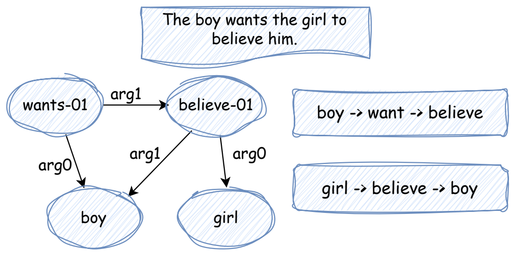

Figure 2: Abstract Meaning Representation for "The boy wants the girl to believe him."

To construct data cases where uncertainty arises from insufficient capability, we instruct the Large Language Model (LLM) to generate answers based on query and gold documents. If the model fails to produce the correct answer (which is already indicated in the documents), we attribute the uncertainty to its insufficient capability. Conversely, if the model successfully generates the correct answer, we construct a new document set by randomly discarding portions of the gold documents and retrieve some new ones. If the model cannot produce the correct answer under the new document set, we classify the uncertainty as stemming from missing documents. It is important to note that different large language models possess varying knowledge and capability boundaries. During evaluation, if a model can generate correct answers based on the original query and provided documents, it is deemed free of uncertainty, and such cases will be excluded from the evaluation.

For uncertainty arising from ambiguity, we directly utilize the ambiguous queries provided in AmbigQA (Min et al., 2020). For the other four datasets, we first transform the queries into Abstract Meaning Representation (AMR) (Shi et al., 2023), where each query is represented by entity nodes and the corresponding relationships between those entities, forming a graph-based structure as shown in Figure 2. Subsequently, we prompt GPT-

4o to introduce ambiguity into the AMR graph by removing modifiers and descriptive words, omitting key information, altering the relationships between nodes, and reorganizing the AMR structure. This method enables the model to better understand the semantic structure of the query, allowing us to provide clearer and more direct instructions for transforming the AMR into an ambiguous query. Then, we convert the AMR into an ambiguous query and generate the corresponding clarifications. If the model fails to answer the ambiguous query but successfully responds to it when provided with the clarification, we categorize the query as ambiguous.

		document	ambiguity	ability
QA	HotpotQA	859	702	141
	AmbigQA	543	537	167
Assistant	ExpertQA	442	397	141
	TechQA	470	683	140
Tool Usage	ToolBench	479	590	144

Table 2: statistics of the benchmark

The statistics of the dataset is shown in Table 2. Additionally, we manually select 50 cases each for queries lacking documentation and those that are ambiguous, as well as 30 cases for instances categorized as lacking ability, from each dataset to construct the benchmark. The remaining cases are used as training data. Consequently, the benchmark comprises a total of $5 \times (50+50+30) = 650$ cases.

4 Preliminary Test

To evaluate the ability to address uncertainty, we instruct the LLM to determine whether it should interact with the retrieval system, consult a user, or utilize Chain of Thought (CoT) reasoning to resolve the uncertainty. If the model opts for CoT, it will generate an answer through a chain of thought reasoning. Conversely, if it chooses to engage with the retrieval system, it will generate a query to retrieve additional documents and then answer the original question based on the results of the interaction. If the model decides to interact with a user, it will formulate inquiries to ask clarifying questions and provide answers based on the received clarifications. We use GPT-4o to simulate the user and provide clarifications based on the inquiries made by the model.

We primarily evaluate the following metrics:

- **Answer Quality (AQ)**: This metric assesses the quality of the answer provided after interaction or using Chain of Thought (CoT). For the HotpotQA and AmbigQA datasets, we employ an LLM as a judge to evaluate correctness. For the other datasets, we score answers based on their usefulness on a scale from 1 to 4; the results shown below are normalized to a range of 0-1.

- **Uncertainty Classification Accuracy (UCA)**: This measures the LLM's capacity to recognize the source of uncertainty, knowing that it should interact with the retrieval system, the user or solve it by CoT.

- **Inquiry Quality (IQ)**: This metric evaluates the quality of the inquiries generated. We compare the query before ambiguity and the gold documents with the actual query and documents provided to the model to derive a gold standard inquiry. We then assess how closely the actual inquiry aligns with the gold inquiry and score it on a scale from 1 to 4 and we normalize it to 0-1 in the paper.

We mainly assess the following models: GPT-4o, Claude-3.5-Haiku, DeepSeek-V3, Qwen2.5-72b-Instruct, Meta-Llama-3-70B-Instruct, Qwen2.5-7b-Instruct, and Mistral-7B-Instruct-v0.2. We use these models to judge the source of uncertainty and generate corresponding inquiries. We aim to evaluate the ability of locating and solving the uncertainty rather than the ability of solving the problem, therefore, to avoid the impact of different perceptions of the question by the models themselves, we use both the evaluated model and GPT-4o to generate answers based on the interaction results, the highest score is considered.

From Table 3, we can observe that the LLM fails to effectively recognize the source of uncertainty and generate corresponding inquiry to solve the uncertainty. DeepSeek-V3 performs best, but only successfully classify about 50% of cases. Those weaker models like Mistral-7b and Qwen2.5-7b fails to effectively recognize the source of uncertainty, even Llama-3-70B shows unsatisfying performance.

From Table 4, we can observe that when faced with uncertainty, LLMs tend to attribute uncertainty to query ambiguity ("ambig") rather than insufficient document support ("doc"), particularly in the less powerful models as indicated by the high recall of "ambig" and low recall of "doc".

		HotpotQA	AmbigQA	TechQA	ExpertQA	ToolBench	avg
AQ	DeepSeek-V3	0.662	0.623	0.788	0.779	0.833	0.737
	GPT-4o	0.631	0.562	0.802	0.806	0.792	0.719
	Claude-3.5-Haiku	0.562	0.512	0.938	0.76	0.767	0.708
	Qwen2.5-72b	0.415	0.563	0.815	0.76	0.813	0.673
	Llama-3-70B	0.377	0.45	0.814	0.742	0.733	0.623
	Mistral-7B	0.315	0.38	0.816	0.765	0.788	0.613
	Qwen2.5-7b	0.338	0.345	0.735	0.756	0.815	0.598
UCA	DeepSeek-V3	0.622	0.545	0.45	0.434	0.713	0.553
	GPT-4o	0.631	0.508	0.447	0.487	0.59	0.533
	Claude-3.5-Haiku	0.652	0.535	0.589	0.426	0.508	0.542
	Qwen2.5-72b	0.545	0.452	0.577	0.442	0.562	0.516
	Llama-3-70B	0.566	0.447	0.408	0.38	0.516	0.463
	Mistral-7B	0.453	0.447	0.407	0.47	0.688	0.493
	Qwen2.5-7b	0.441	0.412	0.385	0.38	0.351	0.394

Table 3: Performance of locating and solving the uncertainty. AQ represents the quality of answer after interaction; UCA is the uncertainty classification accuracy.

	metric	doc	ambig	ability
GPT-4o	precision	0.64	0.43	0.56
	recall	0.24	0.85	0.23
Llama-3-70B	precision	0.56	0.41	0.55
	recall	0.12	0.91	0.24
Qwen2.5-7b	precision	0.3	0.38	0.34
	recall	0.06	0.9	0.13
Mistral-7B	precision	0.45	0.41	0.23
	recall	0.06	0.87	0.16

Table 4: Precision and recall of different uncertainty

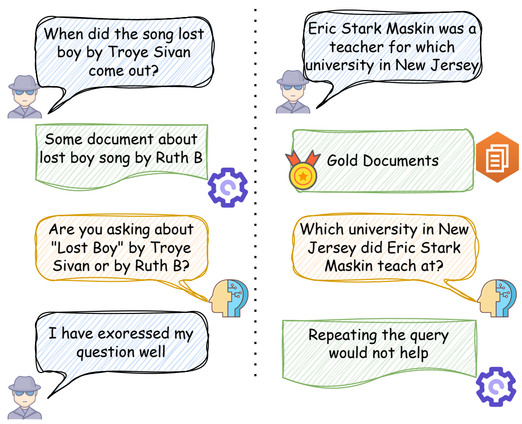

Figure 3: In the left case, the model retrieved some documents about another singer and asks the user to change the query. In the right case, the model simply rephrase the query and wants to retrieve more information.

For models like Qwen2.5-7b, the classification is unbalanced; it recognizes most of the queries as ambiguous and proceeds to interact with the user. We show results of more models and the weighed f1 score of classification is Appendix C.

As illustrated in Figure 3, when presented with a clear query accompanied by noisy documents, the model can become distracted by the noise. It may ask the user to clarify the query, hoping to change the intention of the user so that it can leverage information from the noisy documents to generate an answer. In this context, the model understands why it cannot answer the query, but it places greater trust in the relevance of the documents than in the clarity of the query. Consequently, instead of seeking supplemental documents, the model attempts to align user intention with available information in the noisy documents, ultimately resulting in the unexpected behavior of requiring query rephrasing rather than acquiring more relevant documents.

Moreover, large language models (LLMs) seldom acknowledge that they cannot answer a question due to a lack of capability, as shown in Table 4. In our view, this phenomenon is similar to overconfidence (Xiong et al., 2024; Li et al., 2024b; Xiong et al., 2023); when faced with uncertainty, these models often provide incorrect answers instead of recognizing their limitations with a response such as, "I don’t know." And when instructed to choose the reason for their uncertainty, they also tend to

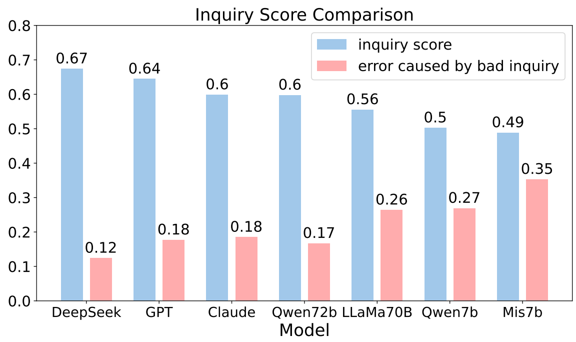

Figure 4: The inquiry score and the percentage of errors caused by bad inquiry (correctly classified but answer incorrectly with low inquiry score)

refuse to acknowledge that it is due to their limited reasoning capacity and blame it to insufficient documents or ambiguity.

As we show in Figure 3, when faced with uncertainty brought by capacity, the model might direct rephrase the query, this can be explained in two ways: 1) the model fails to recognize any lack of documents or ambiguity, so it can only repeat the query again; 2) what confuses the model is the query itself, it fails to effectively understand the query and the given documents, so the query itself is the confusing part.

For the quality of inquiry, we can observe from Figure 4 that powerful models like GPT-4o, Claude-3.5-Haiku, and Qwen2.5-72b are capable of generating meaningful inquiries and effectively resolving uncertainty through these inquiries. Smaller models, such as Qwen2.5-7b, perform worse, the quality of inquiry is not that satisfying showing that smaller models fail to effectively understand the query.

5 Method

In this section, we begin by discussing the use of CoT to identify factors that may be confusing the model and to assess the sources of uncertainty. We then propose that the uncertainty associated with the inquiry is equivalent to that of the query itself. This allows us to directly evaluate the uncertainty by examining the inquiry. And we propose to utilize the uniqueness of the inquiry answer to recognize the source of uncertainty.

5.1 Judge Based on Inquiry Answer

Apparently, judging the source of uncertainty is a difficult task, and less powerful models fail to complete this task; they prefer to regard the query as ambiguous and interact with the user. However, we can also observe from Figure 4 that these models can effectively generate inquiries to interact with their environment, with inquiry scores not significantly lower than GPT-4o and DeepSeek-V3 (Javaji and Zhu, 2024; Qian et al., 2024). A natural approach is to leverage Chain of Thought to identify what is confusing the model, and judge the source of uncertainty afterwards.

Also, we can show that the uncertainty held by the inquiry is actually the same with the query. And the inquiry typically only involve some sub-aspects of the query, it should be easier to identify the source of uncertainty based on the inquiry.

Definition 5.1. *Consider a query x, the corresponding answer y, document d and the clarification to the query c. Let θ be the model and θ^* be the optimal model which can perfectly solve the query x. Then, the uncertainty raised by capacity U_c, by knowledge U_k and by ambiguity U_a*

$$\begin{aligned} U_c &= H(y|x, d, c, \theta), \\ U_k &= H(y|x, c, \theta^*), \\ U_a &= H(y|x, d, \theta^*), \end{aligned} \tag{1}$$

where $H(\cdot)$ stands for entropy

Therefore, U_c is the uncertainty of the model when all information is given, so it is raised by lack of capacity. U_k and U_a is the uncertainty for the optimal model when documents and clarification are missing, they correspond to uncertainty raised by lack of documents and ambiguity. Then, we can show that the inquiry holds similar uncertainty with the query.

Theorem 5.2. *Given a query x and the generated inquiry q, then the uncertainty of q is positively related to the uncertainty of x, then,*

$$\begin{aligned} |U_k(q) - U_k(x)| &\leq -\log p(q^*|x, c, \theta), \\ |U_a(q) - U_a(x)| &\leq -\log p(q^*|x, d, \theta), \\ |U_c(q) - U_c(x)| &\leq -\log p(q^*|x, d, c, \theta), \end{aligned}$$

where q^* is the optimal inquiry generated by θ^*. For lack of ability, the optimal inquiry is the original query.

The theorem posits that if the model generates a meaningful inquiry, the uncertainty held by the inquiry is similar to the original query. Otherwise if the inquiry is meaningless, it shows that the model fails to understand the query well, and it also shows lack of capacity.

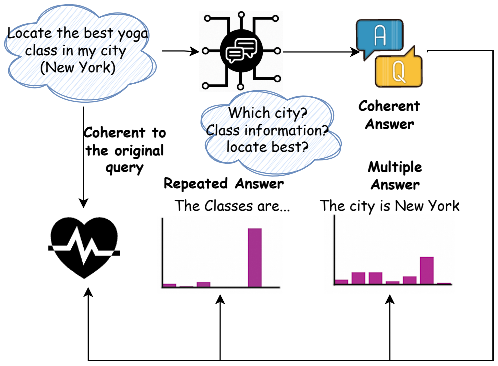

Figure 5: Judge the source of uncertainty based on answer of inquiry

Therefore, we can enhance the performance based on the generated inquiry. If the generated inquiry fails to recognize the confusing part, then we classify it as a lack of capacity, indicating that Chain of Thought is required. Conversely, if the inquiry requires additional documents to solve, retrieval is required otherwise clarification is needed.

To measure the uncertainty held by inquiry, we propose utilizing the answer of inquiry to help the judgment. First, if the uncertainty arises from a lack of capability, the model would merely rephrase the query; thus, the response to the inquiry should appropriately address the original query. Consequently, we can determine whether the uncertainty stems from a lack of capability by evaluating the semantic coherence when the inquiry answer is considered as the answer to the original query. In this way, we can not only recognize cases where Chain of Thought (CoT) is needed but also prevent unnecessary retrieval and clarification.

Also, if the inquiry requires extra retrieval, it typically indicates that the answer points to a definitive objective fact. Conversely, if the query needs further clarification, it suggests the question may have multiple valid subjective answers. To distinguish these scenarios, we designed a verification method inspired by Yadkori et al. (2024): First, we provide the LLM with a logically coherent preset answer to the inquiry. We then instruct the model to generate a new distinct response based on this input. For objective factual questions, the model—lacking prior knowledge—tends to directly repeat the fabricated answer. However, for open-ended subjective questions, the model recognizes the potential for diverse solutions and can still produce novel, reasonable responses even after receiving the preset answer.

For example, consider the query *Locate the best yoga class in my city* and the corresponding inquiry *Which city are you referring to?* If "New York" is provided as a possible answer, the model can easily generate an alternative answer like "London." However, if the inquiry is *Find the yoga classes in New York* and an answer is provided, the model is likely to repeat that answer, indicating it does not know any other answers.

5.2 Inquiry Quality Matters

It is important to note that if the model fails to generate an inquiry of high quality, the advantages of a more concise input may be overshadowed by the drawbacks of a poor inquiry, resulting in suboptimal performance. Therefore, enhancing the quality of the generated inquiry is essential.

Therefore, we propose **InteractDPO**. Vanilla DPO use preference datasets collected ahead of training, the responses in the dataset are usually generated by different LLMs (Rafailov et al., 2024; Qi et al., 2024). Thus, the feedback is usually purely offline. To conduct on-policy training, we first collect some preference datasets, then during training, the trained model generates an inquiry based on the prompt and interact with the retrieval system or the user-GPT to gather more documents or clarification. The model then generates answer based on the interaction. During training, if the trained model successfully generate an inquiry to solve the original query, it will be selected as the chosen inquiry, otherwise the rejected inquiry to conduct on policy DPO training. Compared to directly use LLM to select the chosen-rejected pair like onlineDPO (Qi et al., 2024), InteractDPO provides real feedback and shows better performance.

6 Experiments

To validate the performance of our proposed method, we conduct experiments on the benchmark and various of models.

As shown in table 5, judging by the inquiry and the answer can help to increase the performance, directly judging based on the inquiry can help the performance a lot, it shows greatly improvement on DeepSeek-V3 GPT-4o and Claude. For Qwen2.5-7b, judging based on the inquiry does not help that significantly, this might because the inquiry quality generated is not that satisfying. But we can observe that judge based on the answer still helps, results of more models are shown in Appendix C.

		HotpotQA	AmbigQA	TechQA	ExpertQA	ToolBench	avg
DeepSeek-V3	prompt	0.662	0.623	0.788	0.779	0.833	0.737
	inquiry	0.674	0.623	0.815	0.779	0.837	0.7456
	answer	**0.754**	**0.662**	**0.837**	**0.782**	**0.852**	**0.7774**
GPT-4o	prompt	0.631	0.562	0.802	0.806	0.792	0.7186
	inquiry	0.677	0.623	0.819	0.81	0.854	0.7566
	answer	**0.762**	**0.654**	**0.831**	**0.81**	**0.879**	**0.7872**
Claude-3.5-Haiku	prompt	0.562	0.512	0.808	0.76	0.767	0.6818
	inquiry	0.597	0.618	0.82	0.758	0.84	0.7266
	answer	**0.667**	**0.624**	**0.834**	**0.777**	**0.842**	**0.7488**
Qwen2.5-7b	prompt	0.338	0.345	0.735	0.756	0.713	0.5774
	inquiry	0.315	0.386	0.733	0.764	0.746	0.5888
	answer	**0.408**	**0.392**	**0.76**	**0.759**	**0.749**	**0.6136**

Table 5: Performance of using CoT to judge uncertainty after inquiry generation (inquiry) and judge the uncertainty by inquiry answer (answer), prompt means directly judge the uncertainty source.

	HotpotQA	AmbigQA	TechQA	ExpertQA	ToolBench	avg
GPT-4o	0.762	0.654	0.831	**0.81**	**0.879**	0.7872
vanilla	0.669	0.597	0.784	0.759	0.813	0.7244
SFT	0.732	0.638	0.812	0.784	0.838	0.7608
DPO	0.753	0.642	0.823	0.792	0.845	0.771
onlineDPO	0.746	0.661	0.836	0.784	0.856	0.7766
InteractDPO	**0.779**	**0.669**	**0.836**	0.792	0.861	**0.7874**

Table 6: Performance of InteractDPO

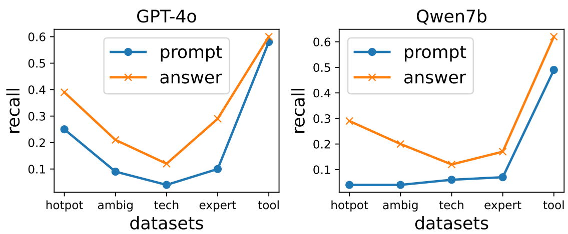

Figure 6: The recall for lack of ability.

Also, as shown in Figure 6, judging based on answer can help to recognize more cases where CoT is required resulting in higher recall, but we also recognize the cases where the model fails to generate a reasonable inquiry as lack of ability, so the precision might be lower, but it helps the performance. The method shows more balanced classification as shown in Appendix C.

For InteractDPO, based on Figure 4, Qwen2.5-7b show great performance when generating inquiry, therefore, we choose the model to conduct further finetuning and enhance its ability of generating high quality inquiries. As we mainly want to enhance and evaluation of generating inquiry, we use the finetuned model to generate inquiry, and use GPT-4o to conduct classification and further answering. Also, we compare our method with SFT, DPO and OnlineDPO, as shown in Table 6, InteractDPO helps the most, the model successfully achieve higher accuracy after finetuning. We also evaluate the performance on uncertainty source identification in Appendix C.

7 Conclusion

In this paper, we discuss the fact that current LLMs fail to effectively judge the source of uncertainty. Models prefers to recognize the query as ambiguous seldomly admit lack of capacity. Then, we propose to judge the source of uncertainty by uniqueness of inquiry answer, to further increase the performance, we propose InteractDPO to help the model generate better inquiry.

Limitations

This paper primarily discusses how current large language models (LLMs) fail to recognize the sources of uncertainty. While we focus on three main categories of uncertainty, these can be further specified. For instance, a lack of documents may correspond to deficiencies in factual knowledge or background information, each requiring different databases for retrieval. Regarding lack of ability, while Chain of Thought (CoT) techniques can address some issues, there are also cases that necessitate the use of methods like Tree of Thought or Monte Carlo Tree Search (MCTS). And there is also various of reasons why the query is ill-defined for example, it could be ambiguous or factually incorrectly or asks for a illegal time. Consequently, there are various sources of uncertainty, each linked to its own solution; however, we only examine the three most common ones.

References

Alfonso Amayuelas, Kyle Wong, Liangming Pan, Wenhu Chen, and William Wang. 2024. Knowledge of knowledge: Exploring known-unknowns uncertainty with large language models. *Preprint*, arXiv:2305.13712.

Tom B Brown. 2020. Language models are few-shot learners. *arXiv preprint ArXiv:2005.14165*.

Vittorio Castelli, Rishav Chakravarti, Saswati Dana, Anthony Ferritto, Radu Florian, Martin Franz, Dinesh Garg, Dinesh Khandelwal, Scott McCarley, Mike McCawley, Mohamed Nasr, Lin Pan, Cezar Pendus, John Pitrelli, Saurabh Pujar, Salim Roukos, Andrzej Sakrajda, Avirup Sil, Rosario Uceda-Sosa, Todd Ward, and Rong Zhang. 2019. The techqa dataset. *Preprint*, arXiv:1911.02984.

Sunhao Dai, Chen Xu, Shicheng Xu, Liang Pang, Zhenhua Dong, and Jun Xu. 2024. Bias and unfairness in information retrieval systems: New challenges in the llm era. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6437–6447.

Yang Deng, Yong Zhao, Moxin Li, See-Kiong Ng, and Tat-Seng Chua. 2024. Don't just say "i don't know"! self-aligning large language models for responding to unknown questions with explanations. *Preprint*, arXiv:2402.15062.

Jiahui Geng, Fengyu Cai, Yuxia Wang, Heinz Koeppl, Preslav Nakov, and Iryna Gurevych. 2024. A survey of confidence estimation and calibration in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6577–6595.

Qiuhan Gu. 2023. 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*, pages 2201–2203.

Bairu Hou, Yujian Liu, Kaizhi Qian, Jacob Andreas, Shiyu Chang, and Yang Zhang. 2023. Decomposing uncertainty for large language models through input clarification ensembling. *arXiv preprint arXiv:2311.08718*.

Hsiu-Yuan Huang, Yutong Yang, Zhaoxi Zhang, Sanwoo Lee, and Yunfang Wu. 2024. A survey of uncertainty estimation in llms: Theory meets practice. *arXiv preprint arXiv:2410.15326*.

Shashidhar Reddy Javaji and Zining Zhu. 2024. What would you ask when you first saw $a^2 + b^2 = c^2$? evaluating llm on curiosity-driven questioning. *arXiv preprint arXiv:2409.17172*.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. 2024. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. *arXiv preprint arXiv:2403.14403*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for "mind" exploration of large language model society. *Preprint*, arXiv:2303.17760.

Jiaqi Li, Yixuan Tang, and Yi Yang. 2024a. Know the unknown: An uncertainty-sensitive method for llm instruction tuning. *arXiv preprint arXiv:2406.10099*.

Moxin Li, Wenjie Wang, Fuli Feng, Fengbin Zhu, Qifan Wang, and Tat-Seng Chua. 2024b. Think twice before trusting: Self-detection for large language models through comprehensive answer reflection. *Preprint*, arXiv:2403.09972.

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. 2024c. Chain of thought empowers transformers to solve inherently serial problems. *arXiv preprint arXiv:2402.12875*.

Chen Ling, Xujiang Zhao, Xuchao Zhang, Wei Cheng, Yanchi Liu, Yiyou Sun, Mika Oishi, Takao Osaki, Katsushi Matsuda, Jie Ji, et al. 2024. Uncertainty quantification for in-context learning of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3357–3370.

Nishanth Madhusudhan, Sathwik Tejaswi Madhusudhan, Vikas Yadav, and Masoud Hashemi. 2024. Do llms know when to not answer? investigating abstention abilities of large language models. *arXiv preprint arXiv:2407.16221*.

Chaitanya Malaviya, Subin Lee, Sihao Chen, Elizabeth Sieber, Mark Yatskar, and Dan Roth. 2024. Expertqa: Expert-curated questions and attributed answers. *Preprint*, arXiv:2309.07852.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. Ambigqa: Answering ambiguous open-domain questions. *arXiv preprint arXiv:2004.10645*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Biqing Qi, Pengfei Li, Fangyuan Li, Junqi Gao, Kaiyan Zhang, and Bowen Zhou. 2024. Online dpo: Online direct preference optimization with fast-slow chasing. *Preprint*, arXiv:2406.05534.

Cheng Qian, Bingxiang He, Zhong Zhuang, Jia Deng, Yujia Qin, Xin Cong, Zhong Zhang, Jie Zhou, Yankai Lin, Zhiyuan Liu, et al. 2024. Tell me more! towards implicit user intention understanding of language model driven agents. *arXiv preprint arXiv:2402.09205*.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Preprint*, arXiv:2305.18290.

Kaize Shi, Xueyao Sun, Li He, Dingxian Wang, Qing Li, and Guandong Xu. 2023. Amr-tst: Abstract meaning representation-based text style transfer. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4231–4243.

Aviv Slobodkin, Omer Goldman, Avi Caciularu, Ido Dagan, and Shauli Ravfogel. 2023. The curious case of hallucinatory (un)answerability: Finding truths in the hidden states of over-confident large language models. *Preprint*, arXiv:2310.11877.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *arXiv preprint arXiv:2212.10509*.

Keheng Wang, Feiyu Duan, Peiguang Li, Sirui Wang, and Xunliang Cai. 2024a. Llms know what they need: Leveraging a missing information guided framework to empower retrieval-augmented generation. *arXiv preprint arXiv:2404.14043*.

Ruobing Wang, Daren Zha, Shi Yu, Qingfei Zhao, Yuxuan Chen, Yixuan Wang, Shuo Wang, Yukun Yan, Zhenghao Liu, Xu Han, et al. 2024b. Retriever-and-memory: Towards adaptive note-enhanced retrieval-augmented generation. *arXiv preprint arXiv:2410.08821*.

Wenxuan Wang, Juluan Shi, Chaozheng Wang, Cheryl Lee, Youliang Yuan, Jen tse Huang, and Michael R. Lyu. 2024c. Learning to ask: When llms meet unclear instruction. *Preprint*, arXiv:2409.00557.

Ziyu Wang and Chris Holmes. 2024. On subjective uncertainty quantification and calibration in natural language generation. *arXiv preprint arXiv:2406.05213*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation. *Preprint*, arXiv:2308.08155.

Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. 2023. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms. *arXiv preprint arXiv:2306.13063*.

Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. 2024. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms. *Preprint*, arXiv:2306.13063.

Yasin Abbasi Yadkori, Ilja Kuzborskij, András György, and Csaba Szepesvári. 2024. To believe or not to believe your llm. *Preprint*, arXiv:2406.02543.

Yongjin Yang, Haneul Yoo, and Hwaran Lee. 2024a. Maqa: Evaluating uncertainty quantification in llms regarding data uncertainty. *arXiv preprint arXiv:2408.06816*.

Yuqing Yang, Ethan Chern, Xipeng Qiu, Graham Neubig, and Pengfei Liu. 2024b. Alignment for honesty. *Preprint*, arXiv:2312.07000.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *Preprint*, arXiv:1809.09600.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Zhangyue Yin, Qiushi Sun, Qipeng Guo, Jiawen Wu, Xipeng Qiu, and Xuanjing Huang. 2023. Do large language models know what they don't know? In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8653–8665, Toronto, Canada. Association for Computational Linguistics.

Tianjun Zhang, Shishir G Patil, Naman Jain, Sheng Shen, Matei Zaharia, Ion Stoica, and Joseph E Gonzalez. 2024a. Raft: Adapting language model to domain specific rag. *arXiv preprint arXiv:2403.10131*.

Tong Zhang, Peixin Qin, Yang Deng, Chen Huang, Wenqiang Lei, Junhong Liu, Dingnan Jin, Hongru Liang, and Tat-Seng Chua. 2024b. Clamber: A benchmark of identifying and clarifying ambiguous information needs in large language models. *arXiv preprint arXiv:2405.12063*.

Ziyuan Zhuang, Zhiyang Zhang, Sitao Cheng, Fangkai Yang, Jia Liu, Shujian Huang, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. 2024. Efficientrag: Efficient retriever for multi-hop question answering. *arXiv preprint arXiv:2408.04259*.

A InteractDPO

In order to improve the ability of locating the uncertainty and generate the corresponding inquiry, we propose **InteractDPO**. Vanilla DPO use preference datasets collected ahead of training the responses in the dataset are usually generated by different LLMs. Thus, the feedback is usually purely offline. Also, different model holds different knowledge, the query might be difficult for model A, but it might be easy for model B. Therefore using dataset generated by one model to train another model is not a good choice.

Also, using a model to judge the quality of inquiry for training like onlineDPO is also not a good choice because the quality of inquiry can hardly be measured because different model may hold different uncertainty when faced with the same query.

To solve this we propose **InteractDPO**. We first collect some preference datasets, then during training, the trained model generates an inquiry based on the prompt and interact with the retrieval system or the user-GPT to gather more documents or clarification. The model then generates answer based on the interaction. If the answer is better than the one generated based on the original query and documents, the inquiry should be a chosen one, otherwise a rejected one.

The preference dataset should contain a prompt which holds some uncertainty, and it should be solvable, which means that there should be an inquiry that can solve the query by interact with the retrieval system or the userGPT. Therefore, we use three different models (GPT-4o, Qwen2.5-7b and Mistral-7b) to generate inquiry based on the query and answer the question after interaction. Then for GPT-4o, we choose those queries that can not be answer correctly without inquiry and can be answered after interaction as chosen. For the same query, those inquiries that fails to answer the query after interaction are considered as rejected.

During training, if the trained model successfully generate an inquiry to solve the original query, it will replace the chosen inquiry, otherwise the rejected inquiry to conduct on policy DPO training.

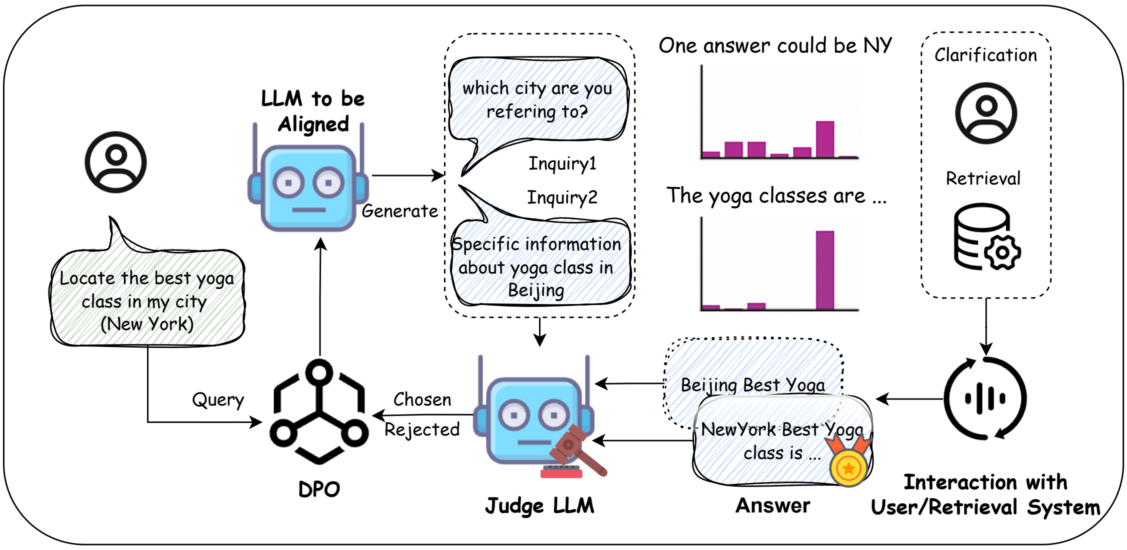

Figure 7: Method Pipeline

B Proof of Theorem

Consider the uncertainty raised by ambiguity.

$$
\begin{aligned}
U_a &= H(y|x, d, \theta^*) \\
H(y|x, d, \theta^*) &= -p(y|x, d, \theta^*) \log p(y|x, d, \theta^*) \\
p(y|x, d, \theta^*) &= p(c|x, d, \theta^*) \cdot p(y|x, d, c, \theta^*)
\end{aligned}
$$

Assumption B.1. *the optimal model θ^* can perfectly solve the problem x with corresponding documents and clarification, which means that $p(y^*|x, d, c, \theta^*) = 1$, y^* is the ground truth answer to query x. And $p(q^*|x, d, \theta^*) = 1$, where q^* is the optimal inquiry.*

In this way

$$
\begin{aligned}
p(y|x, d, \theta^*) &= p(c|x, d, \theta^*) \cdot p(y|x, d, c, \theta^*) \\
&= p(c|x, d, \theta^*) = p(q|x, d, \theta^*) \cdot p(c|q)
\end{aligned}
$$

Therefore,

$$
U_a = H(y|x, d, \theta^*) = H(c|x, d, \theta^*) = H(c|q).
$$

when we generate the inquiry with the optimal model θ^*, the uncertainty of the query is exact the same with the one with the inquiry.

When considering generate the inquiry with the model θ, let $P = p(y|x, d, \theta^*)$, $Q = p(y|x, d, \theta)$, then

$$
\begin{aligned}
H(P) = H(P, Q) - D_{KL}(P||Q) &\geq H(Q) - D_{KL}(P||Q) \\
H(Q) - H(P) &\leq D_{KL}(P||Q)
\end{aligned} \tag{2}
$$

$$
\begin{aligned}
D_{KL}(P||Q) &= \int P(x) \log \frac{P(x)}{Q(x)} \\
&= \int p(y|x, d, \theta^*) \log \frac{p(y|x, d, \theta^*)}{p(y|x, d, \theta)} \\
&= \int p(c|q) \cdot p(q|x, d, \theta^*) \log \frac{p(q|x, d, \theta^*)}{p(q|x, d, \theta)} \\
&= \int p(c|q^*) \log \frac{p(q^*|x, d, \theta^*)}{p(q^*|x, d, \theta)} \\
&= -\log p(q^*|x, d, \theta)
\end{aligned} \tag{3}
$$

$$
\begin{aligned}
D_{KL}^+(Q||P) &= \int_{P(x) \neq 0} Q(x) \log \frac{Q(x)}{P(x)} \\
&= \int p(q^*|x, d, \theta) p(c|q^*) \log \frac{p(q^*|x, d, \theta)}{p(q^*|x, d, \theta^*)} \\
&= p(q^*|x, d, \theta) \log p(q^*|x, d, \theta) - 0 \\
&= -H^+(Q) + H(P) \\
&\geq -H(Q) + H(P)
\end{aligned} \tag{4}
$$

So

$$
\begin{aligned}
H(P) - H(Q) &\leq p(q^*|x, d, \theta) \log p(q^*|x, d, \theta) \\
H(Q) - H(P) &\leq -\log p(q^*|x, d, \theta) \\
|H(P) - H(Q)| &\leq -\log p(q^*|x, d, \theta)
\end{aligned} \tag{5}
$$

It works similarly when face with knowledge uncertainty and lack of capacity, the optimal inquiry for lack of capacity is defined as the original query.

Also, generating the inquiry relates to comprehensively understand and analyze the query and documents, which is a similar task compared to generating the answer, so we assume that $U_c = H(y|x, d, c, \theta) \propto H(q^*|x, d, c, \theta)$

		HotpotQA	AmbigQA	TechQA	ExpertQA	ToolBench	avg
GPT-4o	prompt	0.631	0.508	0.447	0.487	0.59	0.5326
	inquiry	0.573	0.561	0.575	0.469	0.814	0.5984
	answer	0.614	0.588	0.591	0.543	0.785	0.6242
Claude-3.5-Haiku	prompt	0.652	0.535	0.589	0.426	0.508	0.542
	inquiry	0.566	0.564	0.533	0.434	0.697	0.5588
	answer	0.688	0.591	0.517	0.444	0.754	0.5988
DeepSeek-V3	prompt	0.622	0.545	0.45	0.434	0.713	0.5528
	inquiry	0.55	0.556	0.488	0.519	0.719	0.5664
	answer	0.593	0.688	0.515	0.539	0.73	0.613
Qwen2.5-72b	prompt	0.545	0.452	0.577	0.442	0.562	0.5156
	inquiry	0.604	0.564	0.504	0.442	0.694	0.5616
	answer	0.634	0.622	0.597	0.442	0.702	0.5994
Llama-3-70B	prompt	0.566	0.447	0.41	0.38	0.516	0.4638
	inquiry	0.664	0.565	0.479	0.512	0.525	0.549
	answer	0.688	0.505	0.512	0.523	0.576	0.5608
Qwen2.5-7b	prompt	0.441	0.412	0.385	0.38	0.351	0.3938
	inquiry	0.478	0.455	0.423	0.403	0.463	0.4444
	answer	0.55	0.563	0.454	0.469	0.443	0.4958
Mistral-7B	prompt	0.453	0.447	0.408	0.47	0.69	0.4936
	inquiry	0.45	0.61	0.525	0.565	0.412	0.5124
	answer	0.45	0.642	0.553	0.512	0.612	0.5538

Table 7: Classification accuracy for judge based on answer and inquiry. For HotpotQA, the classification performance of judge by inquiry and anawer is worse than by prompt for some model, this is mainly because HotpotQA is a comparably easy dataset, and direct use prompt can have goold result. However, judge by inquiry can better guarantee than every correct classification can result in quality interaction, and can result in better overall answer quality.

C Further Experiments

We conduct further experiments showing the classification accuracy, f1 score and more results on judge based on inquiry and the inquiry answer. Table 7 and 8 shows the classification and f1 score, showing that judge based on the inquiry achieve a better and more balance classification performance. Table 9 shows the result of all models when judge the source of uncertainty based on inquiry and the answer, and we show more results of precision and recall in Table 10.

C.1 Experimental Setup

When conduct training, we use learning rate ranging from $\{3e = 06, 1e-05, 3e-05\}$, and we train the model for 5 epochs. We train the model using LoRA, the rank is set to 64, and the lora targets are q_proj,k_proj,v_proj,o_proj and ffn. the cutoff length is set to 32k, and bf16 training is used.

D Prompts

Prompt to Ambiguate the AMR

```
Gievn a query and the corresponding
Abstract Meaning Representation (AMR),
you should manipulate the AMR to obscure
 it, making it impossible to answer
without further clarification. Make sure
 that the obscured AMR should not change
 the intention of the question, the
obscured AMR should be unanswerable, and
 the obscured AMR should also be a
question rather than a statement. Here
are some possible actions to manipulate
the AMR.

1. Remove certain modifiers and
descriptive words to make some nouns in
the query ambiguous.
2. Delete some key information, making
the query impossible to answer
3. Change the relation between nodes to
make their relationship ambiguous
```

		HotpotQA	AmbigQA	TechQA	ExpertQA	ToolBench	avg
GPT-4o	prompt	0.519	0.436	0.263	0.412	0.496	0.4252
	inquiry	0.421	0.477	0.503	0.386	0.671	0.4916
	answer	0.496	0.444	0.435	0.422	0.666	0.4926
Claude-3.5-Haiku	prompt	0.602	0.508	0.404	0.353	0.47	0.4674
	inquiry	0.55	0.492	0.402	0.375	0.607	0.4852
	answer	0.542	0.489	0.354	0.364	0.64	0.4778
DeepSeek-V3	prompt	0.567	0.476	0.282	0.32	0.665	0.462
	inquiry	0.48	0.44	0.355	0.444	0.633	0.4704
	answer	0.513	0.494	0.372	0.457	0.606	0.4884
Qwen2.5-72b	prompt	0.474	0.389	0.414	0.339	0.499	0.423
	inquiry	0.589	0.533	0.388	0.371	0.629	0.502
	answer	0.577	0.528	0.457	0.379	0.597	0.5076
Llama-3-70B	prompt	0.403	0.297	0.207	0.214	0.447	0.3136
	inquiry	0.617	0.429	0.414	0.445	0.455	0.472
	answer	0.621	0.441	0.459	0.461	0.523	0.501
Qwen2.5-7b	prompt	0.23	0.306	0.212	0.214	0.229	0.2382
	inquiry	0.307	0.359	0.344	0.267	0.37	0.3294
	answer	0.398	0.351	0.333	0.338	0.333	0.3506
Mistral-7B	prompt	0.214	0.273	0.213	0.27	0.426	0.2792
	inquiry	0.29	0.305	0.253	0.268	0.196	0.2624
	answer	0.317	0.336	0.268	0.244	0.326	0.2982

Table 8: Weighed f1 score for classification. For HotpotQA, the classification performance of judge by inquiry and anawer is worse than by prompt for some model, this is mainly because HotpotQA is a comparably easy dataset, and direct use prompt can have goold result. However, judge by inquiry can better guarantee than every correct classification can result in quality interaction, and can result in better overall answer quality.

		HotpotQA	AmbigQA	TechQA	ExpertQA	ToolBench	avg
DeepSeek-V3	prompt	0.662	0.623	0.788	0.779	0.833	0.737
	inquiry	0.674	0.623	0.815	0.779	0.837	0.7456
	answer	**0.754**	**0.662**	**0.837**	**0.782**	**0.852**	**0.7774**
GPT-4o	prompt	0.631	0.562	0.802	0.806	0.792	0.7186
	inquiry	0.677	0.623	0.819	0.81	0.854	0.7566
	answer	**0.762**	**0.654**	**0.831**	**0.81**	**0.879**	**0.7872**
Claude-3.5-Haiku	prompt	0.562	0.512	0.808	0.76	0.767	0.6818
	inquiry	0.597	0.618	0.82	0.758	0.84	0.7266
	answer	**0.667**	**0.624**	**0.834**	**0.777**	**0.842**	**0.7488**
Qwen2.5-72b	prompt	0.415	0.565	0.815	0.76	0.813	0.6736
	inquiry	0.638	0.578	0.813	0.768	0.847	0.7288
	answer	**0.654**	**0.611**	**0.815**	**0.787**	**0.849**	**0.7432**
Llama-3-70B	prompt	0.377	0.45	0.816	0.742	0.733	0.6236
	inquiry	0.462	0.425	**0.823**	0.74	0.735	0.637
	answer	**0.485**	**0.465**	0.812	**0.752**	**0.744**	**0.6516**
Mistral-7B	prompt	0.315	0.38	0.815	0.765	0.788	0.6126
	inquiry	0.646	0.505	0.807	**0.815**	0.779	0.7104
	answer	**0.727**	**0.512**	**0.827**	0.794	**0.789**	**0.7298**
Qwen2.5-7b	prompt	0.338	0.345	0.735	0.756	0.713	0.5774
	inquiry	0.315	0.386	0.733	**0.764**	0.746	0.5888
	answer	**0.408**	0.392	**0.76**	0.759	**0.749**	**0.6136**

Table 9: The answer quality of all models

		prompt			inquiry			answer		
		doc	ambig	ability	doc	ambig	ability	doc	ambig	ability
Precision	GPT-4o	0.64	0.43	0.56	0.54	0.53	0.55	0.54	0.53	0.26
	Claude-3.5-Haiku	0.64	0.45	0.41	0.52	0.56	0.41	0.51	0.55	0.34
	DeepSeek-V3	0.68	0.46	0.51	0.5	0.55	0.46	0.49	0.59	0.4
	Qwen2.5-72b	0.69	0.44	0.44	0.54	0.57	0.42	0.52	0.56	0.31
	Llama-3-70B	0.56	0.41	0.55	0.53	0.48	0.49	0.52	0.48	0.48
	Qwen2.5-7b	0.3	0.38	0.34	0.67	0.4	0.33	0.63	0.4	0.26
	Mistral-7B	0.45	0.41	0.23	0.43	0.42	0.19	0.43	0.37	0.21
Recall	GPT-4o	0.24	0.85	0.23	0.75	0.48	0.18	0.54	0.46	0.34
	Claude-3.5-Haiku	0.39	0.69	0.31	0.77	0.3	0.36	0.7	0.27	0.41
	DeepSeek-V3	0.37	0.81	0.24	0.73	0.38	0.26	0.7	0.37	0.31
	Qwen2.5-72b	0.26	0.79	0.32	0.74	0.4	0.3	0.64	0.41	0.33
	Llama-3-70B	0.12	0.91	0.24	0.51	0.64	0.21	0.5	0.62	0.24
	Qwen2.5-7b	0.06	0.9	0.13	0.12	0.82	0.22	0.15	0.74	0.27
	Mistral-7B	0.06	0.87	0.16	0.44	0.07	0.52	0.42	0.07	0.55

Table 10: Precision and recall of the methods

	HotpotQA	AmbigQA	TechQA	ExpertQA	ToolBench	avg
GPT-4o	0.614	0.588	0.591	0.543	0.785	0.6242
vanilla	0.468	0.563	0.519	0.476	0.622	0.5296
SFT	0.58	0.623	0.523	0.501	0.704	0.5862
DPO	0.621	0.608	0.538	0.496	0.719	0.5964
onlineDPO	0.607	0.627	0.546	0.503	0.71	0.5986
InteractDPO	0.651	0.65	0.554	0.523	0.754	0.6264

Table 11: The uncertainty classification performance of InteractDPO

```
4. Reorganize the structure of the AMR,
make it less clear

The following are some requirements for
the obscured query.

1. The obscured query should still be a
question rather than a statement
2. the obscured query should be similar
to a question that a man would actually
ask rather than some vague question like
 "what is the man's name"
3. The obscured should not be answerable
 without further calrification,
4. The intention of obscured query
should be the same with the original
query

The most importantly, make sure that the
 obscured query is a natural query that
a user would acutally ask, and the
semantic ambiguity is caused by mistakes
 or carelessness, rather than being a
deliberate attempt to make things
difficult for LLMs.

Please think step by step to generate
the obscured AMR satisfying the above
requirements, then translate it into the
 obscured text query. Your output should
 be formatted as Dict{"
step_by_step_thinking": Str(explanation)
, "Obscured Abstract Meaning
Representation (AMR)": Str{AMR}, "
Translated Text Query": Str(obscured
text query)}.

Query: {}
Abstract Meaning Representation (AMR):
{}

Please think step-by-step and generate
your output in json:
```

Prompt to check the result of ambiguity

```
Gievn a query, its obscured version and
clarified query based on the obscured
query, now you need to judge that is the
 obscurity successful. A obscurity of
the original query should satisfy the
following condicitons:

1. The obscured query should still be a
question rather than a statement
2. the obscured query should be similar
to a question that a man would actually
ask rather than some vague question like
 "what is the man's name"
3. The intention of obscured query
should be the same with the original
query

Here we give some examples showing that
the obscure query is a failure,
...

Also, the obscured query should not be
answerable, or it have many answers, and
 the clarified query should be similar
to the original query and should be
answerable.

Therefore, the answer of those query
should satisfy:
1. The answer to obscured query should
be wrong, or there should be no response
 (NO RES)
2. For the obscured query with
clarification, the answer should be the
same or similar to the answer to the
original query

Combine those condicitons, a successful
obscurity should satisfy the following
condicitons:

1. The obscured query should still be a
question rather than a statement
2. the obscured query should be similar
to a question that a man would actually
ask rather than some vague question like
 "what is the man's name"
3. The obscured should not be answerable
, or it have many answers
4. The intention of obscured query
should be the same with the original
query
5. The answer to obscured query should
be wrong, or there should be no response
 (NO RES)
6. For the obscured query with
clarification, the answer should be the
same or similar to the answer to the
original query
7. If the answer to the original query
is NO RES or wrong, then even if the
```

```
answer to the obscured query is wrong
can not ensure that the obscurity is
successful. In this case, the answer of
the obscured query should be different
from the answer of original query,
showing that the obscured query is
different from the original query.

Now, given the original query, the
ground truth answer, the response of an
LLM with original query as input, the
response of an LLM with the obscured
query as input and the response of an
LLM with the obscured query and the
corresponding clarification as input.
All the responses are generated for
multiple times. Please think step by
step and judge that is the obscurity
successful. Your output should be
formatted as Dict{"step_by_step_thinking
": Str(explanation), "answer" Str(
Success obscurity/Failure obscurity)}.

Original Query: {}
Answer to Original Query: {}

Obscured Query: {}
Answer to Obscured Query: {}

Clarified Query: {}
Answer to Clarified Query: {}

Please think step-by-step and generate
your output in json:
```

Prompt to generate gold inquiry

```
Below is a question the corresponding
gold documents to answer the question.
We hide some key information to answer
the question by obscuring the question
or hiding some documents. Your task is
to recognize those missing information
and generate a corresponding inquiry to
gather those information step by step.

We would only provide the query
information or the document information.
 When we provide query information, you
should identify what information is
missing in the actual query compared to
the original query. When we provide
document information, you should
identify which document is missing in
the actual documents.

Now please generate the inquiry for the
following query
Original Query: {}
Gold Document: {}
Actual Query: {}
Actual Document: {}

Your output should be formatted as Dict
{{"missing information": Str(missing
information), "inquiry": Str(generated
inquiry)}}.
Your should strictly format your
response in this format, no extra tokens
should be added.
```

Prompt to evaluate the inquiry

```
Given a question the corresponding gold
documents to answer the question, we
obscure the question or hide some key
documents and generate an inquiry to
gather those missing information. Your
task is to evaluate the quality of the
inquiry.
Evaluation Criteria:

Accurate: Does the inquiry directly
indicate the missing information?
Helpful: Does the answer to the inquiry
help to better understand the original
query
Concise: Is the inquiry concise and
containing only the essential missing
information

Scoring: Rate outputs on a scale of 1 to
 4:
1. Irrelevant: The inquiry is useless,
it simply rewrite the given query
2. Somewhat Relevant: The inquiry is
somewhat relevant to the missing
information, but the inquiry can hardly
gather useful information
3. Basically Relevant: The inquiry asks
something relevant to the missing
information, there is a certain
possibility of obtaining relevant
information by the inquiry.
4. Good: The inquiry directly asks the
missing information, but not concise
enough, there is great possibility that
some useful information would be
gathered.

Original Query: <{}>
Gold Document: <{}>
Actual Query: <{}>
Actual Document: <{}>
Missing Detail and Gold Inquiry: <{}>
Problematic Inquiry: <{}>

Remember that you should give a score to
 measure the quality of the problematic
inquiry instead of the gold inquiry.

You should think step by step and your
output should be formatted as Dict{{"
step by step thinking": Str(explanation)
, "quality of inquiry": 1/2/3/4}}. You
should strictly format your response in
this format, no extra tokens should be
added.
```

Prompt to generate clarification

```
You are an user who asks a question to
the LLM, the query you provided might be
 ambiguous and the LLM asks you for
further clarification by inquiry. You
```

```
need to answer the inquiry based on your
 original intention and the actual query
 you give to the LLM.

Original Intention: {}
Actual Query: {}
Inquiry: {}

Note that you do not know the answer to
your original intention and if the
inquiry involves the answer of the
original intention, please answer with "
This question is beyond scope we can not
 answer your question".

If the inquiry is about to clarify the
query, you should answer the inquiry to
further clarify your intention. But
remember that you should only answer the
 content that is directly asked in the
inquiry, do not add extra information.

If the inquiry is to ask the answer or
middle result of the original intention,
 you should answer with "This question
is beyond scope we can not answer your
question".

Please generate your response strictly
within 50 tokens.
```

Prompt to judge the source of uncertainty

```
One user gives a query and some
documents are retrieved to help answer
the query. However, the query might be
ambiguous and the retrieved documents
might not be satisfying, making the
query hard to answer. Your task is to
identify why the query is hard to answer
.

Question:
{}
Document:
{}

Based on those information. Here are
three kinds of actions you can take,

A: Interact with the retrieval system.
If you need some more factual
information or gather more documents to
answer the query, you should interact
with the retrieval system to get more
information.
B: Interact with the user. If the query
is ambiguous or there exists many
answers, you should interact with the
user to get some clarification.
C: Conducting Chain of Thought. If it
seems that the document information is
adequate and the query itself is not
ambiguous, then the query might need
deeper thinking to solve.

Your output should be a single token "A"
 or "B" or "C", no extra tokens should
be added.
```

Prompt to generate inquiry for further retrieval

```
One user gives a query and some
documents are retrieved to help answer
the query. However, the retrieved
documents is satisfying, making the
query hard to answer. Your task is to
generate an inquiry to gather further
document information to answer the
question.

Question:
{}
Document:
{}

Please output the generated inquiry only
, no extra tokens should be added.
```

Prompt to generate inquiry for clarification

```
One user gives a query and some
documents are retrieved to help answer
the query. However, the query is
ambiguous, making the query hard to
answer. Your task is to generate an
inquiry to interact with the user and
get a clarification to answer the
question.

Question:
{}
Document:
{}

Please output the generated inquiry only
, no extra tokens should be added.
```

Prompt to generate inquiry based on prompt

```
One user gives a query and some
documents are retrieved to help answer
the query. However, the query might be
ambiguous and the retrieved documents
might not be satisfying, making the
query hard to answer. Your task is to
identify why the query is hard to answer
 and generate an inquiry to gather
further information to answer the
question.

Here are some requirements for the
inquiry
1. You should ask for only one question
in the inquiry.
2. Simply describe your question, do not
 add some words like "Could you",
especially you are asksing for document/
API information, because the user can
not provide this information, instead a
retrieval system could. So you should
organize your inquiry as "I need more
information about xxx", "What does xxx
means/refers to", and avoid using words
like "Could you".
3. The inquiry should be concise and
include keywords and it should involve
```

```
limited aspects of the query rather than
 directly asks the query again.
4. Make sure that your inquiry should
only involve some sub-aspects of the
original query and it should be concise
and shorter than the original query.
5. Your inquiry would be directly sent
to the retrieval system or the user for
further clarification, so organize your
inquiry.
6. The retrieval system and the user do
not know the document sent to you, so
organize your inquiry well.

You should only response with the
inquiry and no extra tokens should be
added.

Here are some examples
Question: Are Edward F. Cline and Floyd
Mutrux both screenwriters?
Response: Is Edward F. Cline
screenwriter?

Question: What league did the team that
played home games at a certain stadium
belong to?
Response: what is the name of the
stadium?


Now given the following Query and
documents, Please generate your inquiry.

Query: {}
Documents: {}


Generated Inquiry:
```

Prompt to judge the uncertainty type of the prompt

```
Given a query, an LLM generate an
further inquiry to gather more
information about the query. Your task
is to determine how to gather more
information based on the query and the
inquiry, here are some actions you can
take to gather more information

A: Interact with the retrieval system to
 retrieve more document information.
B: Interact with the user to get further
 clarification about the original query.
C: Conducting Chain of Thought to
thinker more thoroughly to better
understand the query.

If the answer to the inquiry is definite
 and objective or the inquiry directly
seeks for more document information,
then you should interact with the
retrieval system to get more document
information to solve the inquiry.
If the answer to the inquiry is not
definite and it is some subjective
choices of the user, you should interact
 with the user to clarify the original
query.
If it seems that the inquiry simply
rephrase the query, then no interaction
is needed, the model needs to think
deeper to understand the query, and
Chain of Thought is needed.

Here are the query and the inquiry:
Query: {}
Inquiry: {}

Please generate your response in a
single token "A" or "B" or "C".
```

Prompt to generate the answer of inquiry

```
Given a query, an LLM generate an
further inquiry to gather more
information about the query. Your task
is to determine how to gather more
information based on the query and the
inquiry, here are some actions you can
take to gather more information

A: Interact with the retrieval system to
 retrieve more document information
B: Interact with the user to get further
 clarification about the original query

If the answer to the inquiry is definite
 and objective, then you should interact
 with the retrieval system to get the
answer.
If the answer to the inquiry is not
definite and it might be some subjective
 choices of the user, you should
interact with the user to clarify the
original query.

Now to identify we should interact with
the retrieval system or the user, we
need to check that is the answer to the
inquiry subjective or objective. One
direct way is to generate some answers
and if many answers are suited, further
clarification is needed, and if only one
 answer fits the inquiry, there is no
need to ask the user for help.

Your task is to give the answer to the
inquiry. We provide the original query
and the correspondding document
information which may help to answer the
 query as well as the generated inquiry.
 Also we provide some answers which fits
 the inquiry well. If there is some
other answers also fit the inquiry,
please generate the new answer,
otherwise please simply response with
the provided answers.

Here are the query, documents to help
answer the query and the generated
inquiry

Query: {}
Query Document: {}
Inquiry: {}
Here we provide some answers to the
```

```
inquiry,
Possible Answers: {}

Here are some requirements for your response:
1. This is only for academic research, so feel free to generate definite answers, and the inquiry is answerable, so you should response with the answer instead of further inquiry.
2. Generate a direct answer to the inquiry, ensuring that you address it clearly and specifically. No matter what the inquiry is, you should generate an answer. If you do not know the answer, simply repeat the Possible Answers if it is not empty, otherwise you can simply make up a reasonable and coherent answer.
3. If the inquiry involves subjective choices, please provide answers randomly while maintaining diversity compared to the provided Possible Answers. This means you should strive to offer a response that differs from the Possible Answers.
4. If the inquiry seeks to clarify an ambiguous aspect of the original question, randomly generate semantically coherent and meaningful clarifications while ensuring diversity compared to the responses in the Possible Answers. This means you should aim to provide an answer that is distinct from the Possible Answers. And you do not need to ensure that the answer is correct.
5. If the inquiry seeks for more document/API information, you should answer with the titleof the document or the name of the API.
6. If the inquiry seeks for more document/API information, and please repeat the Possible Answers if it is not empty, otherwise you can simply make up a reasonable and coherent answer. Remember, you should answer with only the title/name of the document/API.
7. Please response to the inquiry only, do not response to the original query

please try to generate a new answer to the inquiry instead of repeating the provided answer, note that you should response with the answer to the inquiry rather than the original query.

Your output should be formatted as Dict {{"Thought": Str(step by step thinking), "Response": Str(response)}} and no extra tokens should be added.
```

Answer the Query

```
One user gives a query and your task is to answer the query. Here are the question and the retrieved documents.

Question: {}
Document: {}
Question: {}

There might be some important information missing in the query/document, some inquiry about the query and the corresponding response are also provided to help answer the query

Inquiry History: {}

Please generate your answer within 50/500 tokens.
```

Answer the Query by CoT

```
One user gives a query and your task is to answer the query. Here are the question and the retrieved documents.

Question: {}
Document: {}
Question: {}

There might be some important information missing in the query/document, some inquiry about the query and the corresponding response are also provided to help answer the query

Inquiry History: {}

Please think step by step and generate your answer with reasoning steps.
```