

UNCERTAINTY OF THOUGHTS: UNCERTAINTY-AWARE PLANNING ENHANCES INFORMATION SEEKING IN LARGE LANGUAGE MODELS

Zhiyuan Hu^{1*} Chumin Liu² Xidong Feng³ Yilun Zhao⁴
 See-Kiong Ng¹ Anh Tuan Luu² Junxian He⁵ Pang Wei Koh⁶ Bryan Hooi¹
¹ National University of Singapore ² Nanyang Technological University
³ University College London ⁴ Yale University,
⁵ The Hong Kong University of Science and Technology ⁶ University of Washington

ABSTRACT

In the face of uncertainty, the ability to *seek information* is of fundamental importance. In many practical applications, such as medical diagnosis and troubleshooting, the information needed to solve the task is not initially given, and has to be actively sought by asking follow-up questions (for example, a doctor asking a patient for more details about their symptoms). In this work, we introduce Uncertainty of Thoughts (UoT), an algorithm to augment large language models with the ability to actively seek information by asking effective questions. UoT combines 1) an *uncertainty-aware simulation approach* which enables the model to simulate possible future scenarios and how likely they are to occur, 2) *uncertainty-based rewards* motivated by information gain which incentivizes the model to seek information, and 3) a *reward propagation scheme* to select the optimal question to ask in a way that maximizes the expected reward. In experiments on medical diagnosis, troubleshooting and the ‘20 Questions’ game, UoT achieves an average performance improvement of 57.8% in the rate of successful task completion across multiple LLMs compared with direct prompting, and also improves efficiency (i.e., the number of questions needed to complete the task). Our benchmark and code are publicly released¹.

1 INTRODUCTION

As the capabilities of large language models (LLMs) grow, they are being increasingly deployed in challenging real-world settings involving uncertainty and ambiguity. In particular, recent work aims to develop LLM agents or assistants Xi et al. (2023); Park et al. (2023) that effectively complete tasks in interactive environments, leading to a growing need for LLMs that can *actively seek the information they need* to solve a task by asking questions in conversational settings. For example, in medical diagnosis, patients often do not initially report their symptoms in full detail. In such situations, the ability of the doctor to ask effective questions is crucial, as a successful diagnosis often depends on revealing important details that the patient did not initially provide (Figure 1).

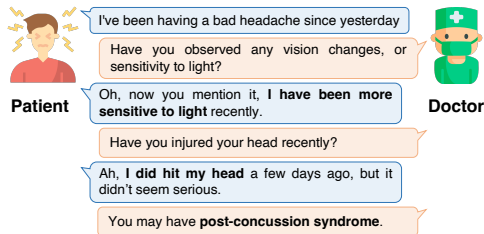


Figure 1: The importance of information seeking in medical diagnosis. The patient initially only complains of a headache, but by asking the right questions, the doctor uncovers the critical information needed for a correct diagnosis.

Recent techniques aim to improve LLMs’ reasoning or planning abilities using the given information rather than urging LLMs to seek information efficiently. For example, Chain-of-Thought (Wei et al.,

*Correspondence to zhiyuan.hu@u.nus.edu

¹<https://github.com/zhiyuanhubj/UoT/>

2022) and Tree-of-Thoughts (Yao et al., 2023) allow LLMs to express middle ‘thoughts’ and reason over them. Unlike these methods, we focus on enabling LLM to ask questions effectively by explicitly guiding the model toward reducing uncertainty, which they do not consider. Thus, they lack effective signals for what questions can better reduce model uncertainty by uncovering critical information.

To enhance LLMs in actively seeking information, we introduce Uncertainty of Thoughts (UoT), a plug-and-play approach that improves LLMs’ abilities to ask useful questions by modeling their own uncertainty. UoT is a principled approach relying on *uncertainty-based rewards motivated by information gain*, which incentivizes a model to seek information in a way that maximally reduces the amount of information it does not know. To utilize these rewards, we develop an *uncertainty-aware simulation framework*, enabling the model to simulate possible future scenarios along with how likely they are to occur. Given these scenarios, we utilize a *reward propagation scheme* to select the optimal question to ask in a way that maximizes the expected reward.

Additionally, most standard benchmarks for LLMs, particularly in question answering, assume that all necessary information to solve a task is provided at the outset, and thus do not evaluate the model’s active information-seeking capabilities. To close this gap, we first introduce a benchmark comprising 5 datasets² on 3 tasks: 20 Questions, a simplified medical diagnosis task, and a basic troubleshooting task. These tasks are designed to measure the model’s ability to ask questions effectively to gather the information they need. For example, the 20 Questions game, also studied by Noever & McKee (2023), requires the model to ask ‘yes’ or ‘no’ questions to determine an unknown object or entity. This scenario serves as a clear and easily analyzed test case, isolating the model’s ability to recognize its own uncertainty, and to ask questions that guide it to the correct answer.

Our work is a step toward LLMs that can effectively operate in settings with high uncertainty and ambiguity, beyond conventional QA settings where all the information needed to solve the task is provided to the model at the outset. To the best of our knowledge, UoT is the first approach for enabling LLMs to ask effective questions by explicitly modeling and seeking to reduce their uncertainty. Our key contributions are as follows:

1. We introduce Uncertainty of Thoughts (UoT), a plug-and-play approach enabling LLMs to explicitly model and seek to reduce their uncertainty. UoT utilizes a principled approach based on an uncertainty-aware framework for simulating possible futures, rewards motivated by information gain, and a reward propagation scheme to select the optimal question to ask.
2. We introduce a benchmark of 3 tasks and 5 datasets, designed to evaluate the ability of LLMs to seek the information they need by asking questions.
3. Experiments show that UoT improves the success rate of multiple LLMs by 57.8% on average compared with direct prompting, achieving top performance on both task success and efficiency. Our benchmark and code are publicly available.

2 METHODOLOGY

2.1 PROBLEM FORMULATION

The problem setting involves two roles: the Questioner and the Answerer, performed by the LLM and a human, respectively. Questioner aims to deduce an unknown piece of information. We formulate this using a *possibility space* Ω , which is the set of all possible options, of which a single element $\omega \in \Omega$, is the *true option* in each given scenario³. For example, in a medical diagnosis setting, Ω is the set of all possible diseases relevant in the context, e.g., $\Omega = \{\text{Bronchitis, Flu, } \dots, \text{Hypertension}\}$, and for each patient, ω is the disease that the patient actually has.

The interaction between the Questioner and the Answerer occurs over multiple turns. For instance, the Questioner may ask, “Do you have a fever?”, to which the Answerer responds, “Yes, I’ve had a high fever for the past two days.” The Questioner then asks another question such as “Have you

²We also incorporate the efforts of prior datasets Srivastava et al. (2022); Xu et al. (2019); Liu et al. (2022); Raghu et al. (2021), through further work and refinement to construct this benchmark. Details are introduced in section 3 Experiments and Appendix 6.7.2.

³Under the measure-theoretic formulation of probability, the *sample point* ω is an element of the *sample space* Ω , and all random variables are defined to be functions of ω . While we conform to this formulation, we try to avoid unnecessary measure-theoretic background for ease of understanding; hence, it is sufficient for readers to understand ω as the ‘true option’ in each scenario.

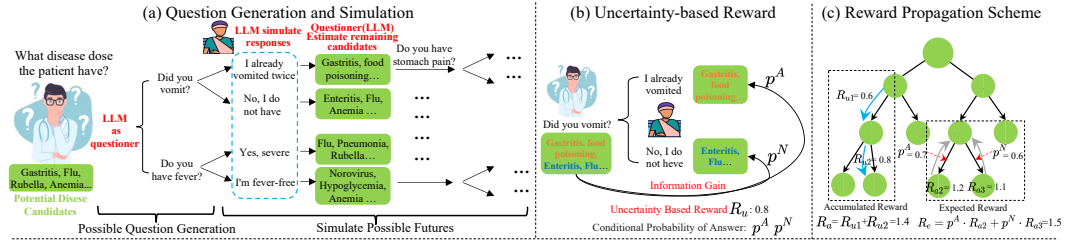


Figure 2: UoT Overview: UoT includes three components: (a) Question Generation and Simulation, where an LLM proposes candidate questions and simulates future scenarios; (b) Uncertainty-based Rewards, measuring the uncertainty reduction from answers to a question, and (c) Reward Propagation computing accumulated rewards R_a over past questions, and expected rewards R_e capturing expected future gains. This ends with choosing questions with the highest expected reward.

vomited?” This exchange continues either until the Questioner correctly determines the final answer, or the conversation reaches a maximum number of turns. At this point, the interaction ends, and the Questioner is *successful* if it has correctly determined the true option ω .

Most of the description of our approach focuses on the *closed set* scenario, in which we assume that the Questioner starts with knowledge of the possibility space Ω , e.g., the set of all possible diseases in medical diagnosis. In our extension section 2.7, we adapt our approach to the *open set* scenario, in which this knowledge is absent. Moreover, as the questioning progresses, we use an LLM to gradually refine this set of possibilities to those that are consistent with the current answers given so far by the Answerer. Define the *current possibility set* Ω_i as the subset of Ω that is consistent with all answers given by the Answerer before the start of the i th interaction step.

As we discuss more later, we focus on applications where answers can be grouped into a small number of semantically distinct categories (in our case, affirmative and negative responses), as this allows us to compute meaningful uncertainty metrics in a simpler way. Conceptually, our framework can straightforwardly be extended to allow for a wider selection of answers.

2.2 UNCERTAINTY OF THOUGHTS: OVERVIEW

As Figure 2 shows, to effectively reduce uncertainty, our UoT method first **generates multiple questions** as candidates to ask, and **simulates possible futures** for each one in the form of a tree structure. Next, **uncertainty-based rewards**, motivated by information gain, are used to assess the questions within the simulation. Finally, a **reward propagation scheme** is used to compute the expected reward from asking each candidate question, allowing us to select the one with highest expected reward, to ask the Answerer.

2.3 QUESTION GENERATION AND SIMULATION

UoT starts by using an LLM to generate several candidate questions, then simulates future scenarios for each one. This simulation process allows us to measure how much information we can expect to gain in the next few steps from each question, and thus to choose the most suitable question.

Question Generation Recall that our setting involves sequential interactions between a Questioner (e.g., a chatbot) and an Answerer (e.g., a human patient). During the i th interaction step, the Questioner generates candidate questions, then selects one of these to ask, denoted as q_i .

To generate candidate questions to ask, UoT uses two inputs: (1) the *history of past interactions* $h_i = \{q_1, a_1, q_2, a_2, \dots, q_{i-1}, a_{i-1}\}$, comprising the sequence of past questions and answers; and (2) the *current possibility set* Ω_i . These two inputs are combined to form a prompt that includes instructions explaining the nature of the task (e.g., how the 20 Questions game works), provides the current history h_i and the current possibility set Ω_i , and asks an LLM to generate m candidate next questions, conditioned on the previous context. This prompt, denoted as $\text{Prompt}_{\text{gen}}(h_i, \Omega_i)$, is fed to our generator LLM_{gen} , which then generates m candidate questions, denoted $q_i^1, q_i^2, \dots, q_i^m$:

$$q_i^1, q_i^2, \dots, q_i^m = \text{LLM}_{\text{gen}}(\text{Prompt}_{\text{gen}}(h_i, \Omega_i)) \tag{1}$$

Multistep Simulation As shown in Figure 2 (a), the Question Generation stage generates candidate questions such as $q_i^1 =$ “Did you vomit?” Next, during Simulation stage, for each such generated

candidate question, we simulate possible futures for a few steps, forming a tree of possibilities. This process enables us to compute rewards for each question, helping us to decide which question to ask.

Each node of the tree can be one of two types: *Answerer Nodes* where it is the Answerer’s turn to answer a question, and *Questioner Nodes* where it is the Questioner’s turn to ask a question. At the root, a question has just been asked (e.g., q_i^1), so the root is an Answerer Node. Next, we explain how to construct tree by recursively expanding (or ‘branching’) each node to construct its children, i.e., starting from the root, then proceeding to its children, and so on.

- At each **Answerer Node**, a question has just been asked. Next, we need to further ‘branch’ the tree based on the possible answers to the current question. Rather than allowing completely open-ended answers, we instead focus on affirmative and negative responses⁴, as this allows us to compute meaningful uncertainty metrics, as we discuss later. Hence, we branch the node into two children, corresponding to affirmative and negative answers.
- At each **Questioner Node**, we prompt an LLM to generate m questions using the current history and possibility set, in the same way as in the Question Generation step. Note that while the generation procedure is similar, the purpose is different: the Question Generation step generates *candidate* questions to select from, while here we are generating *simulated* questions to form a tree for the purpose of evaluating the current question. The resulting m generated questions are added to the tree as children of the current node.

In this way, we recursively generate nodes of the tree, stopping when we reach a fixed number of levels (i.e., depth). While generating this tree, we also recursively compute the *current possibility set* Ω_v at each node v . Specifically, let h_v be the current conversation history up to node v , combining both the actual conversation history h_i and the simulated conversation up to node v . Then the current possibility set at this node, denoted Ω_v , is the subset of the possibility space consistent with h_v . At the root, the current possibility set is only limited by the actual conversation history, i.e., Ω_i . Then, as we proceed over the simulated tree, note that the current possibility set only changes at Answerer nodes, when an answer is added to the current history. Hence, at each Answerer node v , we prompt a new LLM (an ‘Answerer Simulator’ LLM_{ans}), to determine the further subset $\Omega_v^A \subseteq \Omega_v$ for which the answer to the current question is affirmative, and the corresponding $\Omega_v^N = \Omega_v \setminus \Omega_v^A$ for which the answer is negative.⁵ This allows us to recursively compute the possibility sets of the children of v (which themselves correspond to the affirmative and negative answers).

$$\Omega_v^A, \Omega_v^N = \text{LLM}_{\text{ans}}(\text{Prompt}_{\text{ans}}(h_v, \Omega_v)) \quad (2)$$

In this way, we can recursively compute the possibility set on each node of the tree.

2.4 UNCERTAINTY-BASED REWARD CALCULATION

To develop suitable information-seeking approaches, a critical question is *how to evaluate the effectiveness of a question, i.e., its contribution to reducing uncertainty*. To address this, we turn to information theory, specifically the concept of *information gain*, which measures the amount by which uncertainty decreases after a particular observation. To reward information-seeking behavior, we assign rewards to questions based on how much they reduce the model’s uncertainty about the unknown random variable. These reward signals are used by our UoT framework to determine which question to select, to maximize the reduction of uncertainty.

Entropy. Entropy and information gain are well-known concepts in information theory Shannon (1948). In our work, we use these concepts to measure how much information is gained (or equivalently, how much uncertainty is reduced) by asking a question, to formulate our rewards. Entropy measures the level of uncertainty in a random variable: higher entropy indicates greater uncertainty. The entropy of a discrete random variable X taking values x_1, \dots, x_n is:

$$H(X) = - \sum_{i=1}^n p(x_i) \log p(x_i) \quad (3)$$

Since our goal is to reduce the uncertainty in the unknown $\omega \in \Omega$, we use entropy to measure this uncertainty. Formally, let $\Omega = \{\omega_1, \dots, \omega_n\}$, and we define an additional set of arbitrary real

⁴As shown Figure 2 (a), for question ‘Did you vomit?’, possible affirmative responses include ‘yes’ or ‘I already vomited twice’, while negative responses could be ‘no’ or ‘I don’t have’.

⁵In practice, allowing overlap between Ω_v^A and Ω_v^N may be more realistic. However, in this work, we consider only the simplified scenario where they are disjoint.

numbers $\mathcal{X} = \{x_1, \dots, x_n\} \subseteq \mathbb{R}$ which we will associate with each of these possibilities. Define a random variable $X : \Omega \rightarrow \mathcal{X}$ such that $X(\omega_i) = x_i$. Intuitively, X is a discrete random variable that takes the value x_i if the i th possibility is true, i.e., if $\omega = \omega_i$. X serves to capture our uncertainty about ω , since observing X is equivalent to observing the true option ω . As a simple example, suppose our possibility space is $\Omega = \{\omega_1, \omega_2, \omega_3\}$; we accompany these with real numbers x_1, x_2, x_3 , and have a distribution for our random variable X reflecting prior beliefs over these possibilities: e.g., $p(x_1) = 0.2$, $p(x_2) = 0.3$, $p(x_3) = 0.5$. Conceptually, our framework allows for any prior probability distribution over the possibilities (i.e., $p(x_i)$), but in our experiments, we assume a uniform distribution over them due to the lack of an informative prior.

Before asking any questions, our uncertainty about the unknown ω is given by $H(X)$, as in Eq. equation 3. At any node v of the trees described in the previous section, recall that we have a conversation history h_v which contains some answers given by the Answerer. This history limits the current possibility set to those in $\Omega_v \subseteq \Omega$, thereby reducing our uncertainty. We model this using notion of *conditional probability on an event*, with $\Omega_v \subseteq \Omega$ allowing conditioning on Ω_v .

$$p(x_i|\Omega_v) = p(x_i)/p(\Omega_v) \quad \forall i \text{ such that } \omega_i \in \Omega_v \quad (4)$$

where $p(\Omega_v)$ is the sum of probabilities of the elements in Ω_v . To illustrate, we continue from the earlier example, where $p(x_1) = 0.2$, $p(x_2) = 0.3$, $p(x_3) = 0.5$. If the conversation history h_v at node v is only consistent with x_1 and x_2 , i.e., $\Omega_v = \{\omega_1, \omega_2\}$, we can adjust probability distribution by conditioning: e.g., the adjusted probability of x_1 is $p(x_1)/p(\Omega_v) = 0.2/(0.2 + 0.3) = 0.4$.

Next, to quantify the uncertainty at node v , note that since X is conditionally distributed based on $p(\cdot|\Omega_v)$, the entropy of this distribution is:

$$H_v(X) := \sum_{i:\omega_i \in \Omega_v} p(x_i|\Omega_v) \log p(x_i|\Omega_v) \quad (5)$$

Intuitively, $H_v(X)$ is the remaining uncertainty in X at node v (i.e., after observing the history h_v).

Information Gain at a Node We now quantify the uncertainty reduction when receiving answers at an Answerer node v . Recall that the answer given at v partitions Ω_v into two disjoint subsets: $\Omega_v = \Omega_v^A \cup \Omega_v^N$, where Ω_v^A and Ω_v^N are the subsets of possibilities resulting in affirmative and negative answers respectively to last asked question. Given an affirmative answer, entropy reduces to:

$$H_v^A(X) := \sum_{i:\omega_i \in \Omega_v^A} p(x_i|\Omega_v^A) \log p(x_i|\Omega_v^A) \quad (6)$$

We define $H_v^N(X)$ analogously for negative answers. Let $p_v^A = p(\Omega_v^A)/p(\Omega_v)$ and $p_v^N = p(\Omega_v^N)/p(\Omega_v)$ be the conditional probabilities of affirmative and negative answers at node v . To compute the expected entropy after receiving the answer at node v , since we have a p_v^A probability of receiving an affirmative answer and p_v^N of a negative answer, the expected entropy is:

$$p_v^A \cdot H_v^A(X) + p_v^N \cdot H_v^N(X) \quad (7)$$

As such, the expected information gain at node v is the difference in entropies before and after receiving the answer:

$$IG_v(X) := H_v(X) - p_v^A \cdot H_v^A(X) - p_v^N \cdot H_v^N(X) \quad (8)$$

We can simplify this: as proven in Appendix 6.1, the above equation reduces to:

$$IG_v(X) = -p_v^A \log p_v^A - p_v^N \log p_v^N \quad (9)$$

This represents the expected reduction of uncertainty in X when receiving an answer at node v . Note that it has an entropy-like expression, and is therefore nonnegative.

Reward Formulation A natural approach would be to define the reward function $R_u(v)$ at node v as the information gain $IG_v(X)$: that is, the reward from the question at node v is the expected information gain $IG_v(X)$ from receiving its answer. In practice, we find that a slightly modified function $\widetilde{IG}_v(X)$ is preferable. In particular, we find that $IG_v(X)$ does not result in sufficiently sharp differences in reward over the typical ranges we encounter. Hence, we introduce an additional hyperparameter $\lambda \geq 0$ which helps to sharpen the rewards using a scaling approach:

$$R_u(v) = \widetilde{IG}_v(X) := \frac{-p_v^A \log p_v^A - p_v^N \log p_v^N}{1 + \lambda^{-1} |p_v^A - p_v^N|} \quad (10)$$

This definition ensures that $R_u(v)$ falls within the range $[0, 1]$, providing a normalized and consistent reward to measure uncertainty reduction. The reward function reaches its maximum when the subsets Ω_v^A and Ω_v^N have equal probability, reflecting the maximum reduction in uncertainty. It reaches its minimum when one of the subsets has zero probability, indicating no reduction in uncertainty. Appendix 6.5 plots the reward function curve across values of p_v^A and p_v^N .

2.5 QUESTION SELECTION VIA REWARD PROPAGATION

Single-step rewards often fall short in dynamic settings as they only consider immediate impact, overlooking long-term effects. To overcome this, our method employs a reward propagation scheme over simulation trees. Specifically, we will define ‘accumulated rewards’ which accumulate rewards over multiple steps of the simulations, capturing the effectiveness of past decisions. These are used to calculate ‘expected rewards,’ indicating the expected information from the questions. The rewards then guide the selection of candidate questions.

Accumulated Reward We first define the accumulated reward at each node v , which accumulates the rewards at v and all its ancestors on the tree, defined recursively as:

$$R_a(v) := R_u(v) + \begin{cases} 0 & v \text{ is root} \\ R_a(\text{Parent}(v)) & \text{otherwise} \end{cases}$$

Here $R_u(v)$ is the uncertainty-based reward at node v defined in Eq. equation 10, and $R_a(\text{Parent}(v))$ is the accumulated reward of the parent of v . We compute these accumulated rewards by starting at the root and propagating down to the leaves. Intuitively, the accumulated reward at each leaf node represents the total reward we end up with at the end of the conversation at that node.

Expected Reward Next, we compute the expected reward for each node $R_e(v)$, which represents the expected total value of rewards received on expectation on a node and all its descendants on tree:

$$R_e(v) := \begin{cases} R_a(v) & \text{if } v \text{ is a leaf; otherwise:} \\ p_v^A R_e(v^A) + p_v^N R_e(v^N) & \text{if } v \text{ is an Answerer Node} \\ \frac{1}{m} \sum_{w \in \text{Children}(v)} R_e(w) & \text{if } v \text{ is a Questioner Node} \end{cases}$$

For the case where v is an Answerer Node, recall that p_v^A and p_v^N are the conditional probabilities of affirmative and negative answers at node v , defined in section 2.4. v^A and v^N are its children, corresponding to the affirmative and negative answers. For the case where v is a Questioner Node, we assign equal probability to the m questions asked from this node. In this way, we propagate the expected rewards from the leaves up to the root, allowing us to compute the expected gain at the root (i.e., the expected reward for that candidate question).

Determining the Optimal Question Finally, to decide the question to ask, we select the question with highest expected reward (and therefore, the highest expected information gain, considering both immediate and future information gains):

$$q_i = \arg \max_{n=1} R_e(q_i^n) \quad (11)$$

2.6 UoT SUMMARY

UoT first generates candidate questions $q_i^1, q_i^2, \dots, q_i^m$ based on the history h_i and current possibility set Ω_i . Then, we conduct multistep simulation to generate a tree for each candidate question q_i^n . Next, we compute the uncertainty-based rewards $R_u(v)$, and propagate over the trees to compute accumulated reward $R_a(v)$ and expected reward $R_e(v)$. Lastly, the optimal question q_i^n with highest expected reward will be selected as q_i to interact with the Answerer.

2.7 EXTENSIONS AND DISCUSSION

Pruned UoT. To enhance efficiency during simulation, pruning akin to Beam Search can be employed when constructing the simulation trees, which limits the number of paths to explore over the tree to a predetermined size.

Open Set UoT. Recall that in the closed set scenario, the Questioner starts with knowledge of the possibility space Ω . In practice, the possibility space is often unknown, resulting in the open set

setting. To adapt UoT to this case, we prompt the Questioner to initialize the possibility space Ω and then reinitialize the possibility set Ω_i according to current history h_i . Following this, the rest of UoT proceeds unchanged.

3 EXPERIMENTS

3.1 EXPERIMENTAL SETUP

Models We test various LLMs to evaluate the generality of our method, including **Llama 2-70B-Chat** Touvron et al. (2023), **Cohere** Cohere (2023), **PaLM 2** Anil et al. (2023), **Claude 2** Anthropic (2023), **GPT-3.5-turbo** OpenAI (2023a) and **GPT-4** OpenAI (2023b).

Baselines **Direct Prompting in Open-Set (DPOS)** prompts an LLM directly to generate the next response, without using information about the possibility set. **Direct Prompting in Closed-Set (DPCS)** At the start of the interaction, the LLM is provided with full information about the possibility set. The subsequent baselines also follow this setting. **Planning Prompting (PP)** is motivated by Wang et al. (2023). We leverage another LLM to plan the future and, consequently, determine the question to ask. **Chain-of-Thought (CoT)** Wei et al. (2022) enhances LLMs’ complex reasoning via intermediate steps. **CoT-SC (Self-Consistency)** Wang et al. (2022), an ensemble method, explores multiple reasoning paths. We standardize sampling counts for fair computational cost comparison with other methods. **Reflexion** Shinn et al. (2023) enables the agent to propose an action, and uses self-evaluation-based rewards to assess whether the agent should reflect to develop a new idea. **Tree-of-Thoughts (ToT)** Yao et al. (2023) allows LMs to navigate decisions through a tree of reasoning paths. We analyze two versions: **Original-ToT**, following standard question generation and evaluation, and **Adapted-ToT**, incorporating heuristic insights to focus on space-halving questions. Tree depths were aligned with UoT simulation steps for equitable comparison. Hyperparameters and detailed experimental settings are in Appendix 6.7.1, and prompts in Appendix 6.9.

Table 1: Results from three different scenarios, assessing Success Rate (SR), Mean Conversation Length in Successful Cases (MSC), and Mean Conversation Length (MCL).

Model	Method	20 Questions						Medical Diagnosis						Troubleshooting FloDial		
		20Q in BIG-bench			Common			DX			MedDG			SR↑	MSC↓	MCL↓
		SR↑	MSC↓	MCL↓	SR↑	MSC↓	MCL↓	SR↑	MSC↓	MCL↓	SR↑	MSC↓	MCL↓	SR↑	MSC↓	MCL↓
Llama2-70B	DPOS	6.9	12.0	19.5	1.80	11.0	19.8	13.4	3.1	4.8	23.7	3.4	4.6	11.1	15.1	19.5
	DPCS	17.2	13.5	18.9	6.31	12.0	19.7	29.8	3.0	4.4	28.0	3.5	4.6	24.2	14.5	18.7
	UoT	20.7	13.2	18.6	10.8	15.6	19.5	51.9	1.8	3.4	33.9	1.4	3.8	31.4	15.8	18.7
Cohere	DPOS	3.45	15.0	19.8	1.80	14.0	19.9	19.8	3.7	4.7	25.0	3.6	4.7	16.3	16.7	19.5
	DPCS	6.90	12.0	19.4	1.80	12.5	19.8	35.6	3.3	4.4	33.3	4.0	4.7	27.5	16.3	19.0
	UoT	34.3	8.50	16.0	16.2	11.7	18.6	45.5	2.6	3.9	75.7	2.7	3.3	41.4	8.7	15.3
PaLM 2	DPOS	37.9	13.5	17.5	35.1	14.4	18.0	7.69	3.9	4.9	11.3	4.0	4.9	22.6	15.2	19.0
	DPCS	51.7	13.2	16.5	53.1	13.9	16.8	7.92	3.4	4.9	34.0	4.4	4.8	30.1	15.0	18.5
	UoT	72.4	7.0	10.6	62.1	12.5	15.3	75.0	2.1	2.8	80.7	2.2	2.7	48.4	7.6	14.0
Claude2	DPOS	48.3	9.8	15.1	29.7	13.8	18.2	45.2	3.0	4.1	60.7	4.1	4.5	39.7	14.3	17.7
	DPCS	72.4	11.6	13.9	43.2	13.8	17.3	97.1	2.4	2.5	83.0	4.3	4.4	42.9	15.7	18.2
	UoT	75.9	5.1	8.69	61.3	9.8	13.7	98.0	2.3	2.4	88.3	2.7	2.9	52.6	6.3	12.8
GPT-3.5	DPOS	36.0	12.6	17.3	32.6	14.6	18.2	18.8	3.5	4.7	25.0	3.5	4.6	19.4	12.3	18.5
	DPCS	44.8	13.2	17.0	40.0	14.8	17.8	49.5	2.7	3.3	42.3	3.8	4.5	22.6	13.3	18.5
	UoT	51.7	5.3	12.4	44.2	10.9	16.0	92.1	2.1	2.4	81.3	2.4	2.9	67.1	6.9	11.2
GPT-4	DPOS	37.9	14.0	17.7	48.6	14.0	17.1	44.2	3.5	4.9	45.7	4.2	4.6	38.4	13.0	17.3
	DPCS	48.3	11.1	15.7	50.5	13.1	16.5	91.3	3.0	3.3	72.3	4.2	4.4	43.7	13.4	17.1
	PP	58.6	14.0	16.5	38.7	14.9	18.0	58.6	2.5	3.5	62.3	3.8	4.3	39.2	14.2	17.7
	CoT	27.6	13.1	18.1	20.7	16.0	19.2	33.7	3.7	4.4	20.0	3.8	4.3	32.8	10.1	16.8
	CoT-SC	37.9	13.5	17.6	55.1	14.0	16.7	48.5	3.6	4.3	26.7	4.2	4.8	42.5	11.0	16.2
	Reflexion	67.9	9.80	13.1	67.6	12.0	14.6	52.5	3.7	4.3	30.3	4.0	4.7	28.6	11.5	17.8
	Original-ToT	34.5	14.3	18.1	28.8	15.5	18.7	70.3	3.3	3.8	60.3	3.2	3.9	40.4	11.6	16.6
	Adapted-ToT	41.4	10.5	16.1	42.6	12.2	16.1	92.1	1.9	2.2	78.0	3.0	3.4	60.3	8.2	12.9
	Pruned UoT	65.5	8.7	12.6	62.2	10.8	14.3	92.1	1.9	2.1	83.3	2.7	3.1	63.2	8.2	12.5
	UoT	79.3	7.0	9.7	71.2	10.8	13.5	97.0	2.0	2.1	88.0	2.6	2.9	67.3	7.8	11.8

Scenarios and Datasets **20 Questions** is a classic game where the *answerer* thinks of a word or item, and the *questioner* asks up to 20 yes-or-no questions to guess it. We use two distinct datasets, namely **20 Questions (20Q)** in BIG-bench Srivastava et al. (2022), and **Common** (collected by us, refer to Appendix 6.7.2 for more details), including 29 and 111 items separately. In this scenario, the maximal turns is set to 20. In **Medical Diagnosis**, the doctor needs to ask questions to patients about their symptoms, to determine an accurate diagnosis. We use two datasets: **DX** Xu et al. (2019), with 104 doctor-patient dialogues and 5 diseases in test set, and **MedDG** Liu et al. (2022), with over 17K conversations and 15 type diseases, from which we manually select 350 high-quality test samples for evaluation. Both datasets have a maximum of 5 turns. **Troubleshooting** is a scenario where a customer support technician interacts with customers to identify and resolve faults or issues within computer systems, electronic devices, machinery, or other complex systems. Raghu et al. (2021)

introduce FloDial with 894 dialogues, containing 153 faults. We evaluate using a maximum of 20 turns. The answerer is simulated by GPT-4, prompted with the ground truth information for a case (e.g., the patient’s ground truth disease, self-report, and conversation). For more details, refer to Appendix 6.7.2. We provide examples of these scenarios in Appendix 6.8.

Evaluation Metrics To measure efficacy and efficiency, we use: **Success Rate (%)**: $SR = S/T$, where S is the number of successful cases, and T is the total number of cases; **Mean Conversation Length in Successful Cases**: $MSC = R_s/S$, where R_s is the total rounds in successful cases; **Mean Conversation Length**: $MCL = R/T$, where R is the total rounds in all cases. **MCL** measures efficiency based on the resources used in both successes and failures.

3.2 PERFORMANCE

20 Questions (20Q) As illustrated in Table 1, for all types of LLMs, those equipped with UoT outperform the baselines in both DPOS and DPCS settings. Among the methods used on GPT-4 to enhance planning and reasoning, CoT and PP show inferior performance even compared to GPT-4 alone. UoT achieves the highest success rate, surpassing the second-best Reflexion by an average of 7.5%. It’s more efficient than Reflexion, requiring 2 fewer rounds on average in successful cases.

Medical Diagnosis (MD) UoT outperforms baselines in simplified medical diagnostics, achieving a 97.0% success rate on the DX dataset with GPT-4. On the MedDG dataset, UoT on PaLM 2 and GPT-4 achieve success rates of 80.7% and 88.0%. UoT significantly reduces conversation length, with average MSC of 2.0 on GPT-4 for DX, lower than 3.5 and 3.0 for DPOS and DPCS methods.

Troubleshooting (TB) UoT similarly achieves the highest SR of 67.3%, and the lowest MSC of 7.8. With UoT, GPT-3.5 boosts its SR from 22.6% to 67.1% in closed set scenarios.

Overall Performance On average, UoT enhances the success rate by 57.8% compared to DPCS across 5 datasets and 6 different LLMs, including open source and commercial models. Notably, success rate increases 102.8% for Cohere. Furthermore, UoT outperforms CoT-SC by 38.4% and Reflexion by 31.2%. Even compared to tree structure methods like Original-ToT and Adapted-ToT, UoT still shows superior performance with gains of 33.7% and 17.7% respectively. Additionally, Pruned UoT, our pruning method to improve efficiency, outperforms Adapted-ToT by 10.4%.

3.3 ANALYSIS

3.3.1 COMPARING MODEL PERFORMANCE AT EQUAL COMPUTATIONAL EFFICIENCY

Here, we compare performance across approaches with similar computational cost, in terms of token consumption. To do so, we first prune our UoT as described in section 2.7. Secondly, we expand the exploration depth of Adapted-ToT method to bring its token cost in line with that of UoT.

As shown in the top half of Table 2, the Pruned UoT model, despite its reduced efficacy compared to UoT, still outperforms ToT and other methods. Also, the bottom part of Table 2 shows that even when increasing the depth of Adapted ToT (Adapted-ToT ($D = 4$)) to match the token cost of UoT ($D = 3$), it still underperforms compared to UoT.

Table 2: Average success rates for 20Q, MD, and TB datasets at comparable efficiency, measured by GPT-4 token use. k is sampling count, D is tree depth.

Method	Tokens	20Q	MD	TB
CoT-SC($k = 33$)	4.6k	46.5	37.6	42.5
Orig-ToT($D = 3$)	4.5k	31.7	65.3	40.4
Adapt-ToT($D = 3$)	4.5k	37.4	85.1	60.3
Pruned UoT($D = 3$)	4.7k	64.8	88.4	63.2
Adapt-ToT($D = 4$)	9.3k	54.4	86.7	63.7
UoT($D = 3$)	9.2k	75.3	92.5	66.0

3.3.2 EFFECTIVENESS OF UNCERTAINTY REWARDS

To further demonstrate the effectiveness of our uncertainty-based reward, we compare it with the self-evaluation reward used in the original ToT based on GPT-4 model. We implement the uncertainty-based reward in place of the self-evaluation reward in ToT, creating a variant we call ToT (+UR). The results, as shown in left side of Figure 3, indicate that our reward significantly enhances planning efficacy by an average of 9.3%. Additionally, we use the heuristic self-evaluation reward in Adapted-ToT to replace our current uncertainty-based reward in UoT, a variant we refer to as UoT (-UR). This change results in a performance decrease shown in the right part of Figure 3, further validating the effectiveness of our uncertainty-based reward. Moreover, the performance of UoT (-UR) still surpasses that of Adapted-ToT illustrated in Table 1,

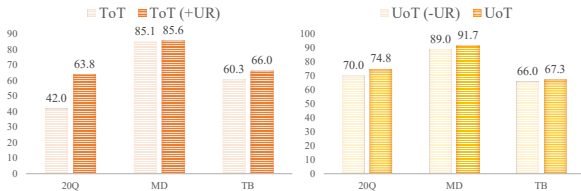


Figure 3: Success rate comparison between Adapted-ToT and Adapted-ToT using uncertainty reward, and between UoT and UoT without uncertainty reward.

Table 3: Averaged Success rate comparison over datasets on open set setting by using GPT-4 in 20Q, MD, and TB tasks.

Method	20Q	MD	TB
DPOS	43.3	45.0	38.4
Open set UoT	46.0	60.9	42.5

3.3.3 CASE STUDIES

As demonstrated in Figure 4 of Appendix 6.2, compared to direct prompting, UoT asks questions that better reduce uncertainty and narrow down candidates, rather than overly specific questions that yield limited information. Moreover, once our method acquires initial information (e.g., stomach pain), it generates more targeted questions (related to digestive issues), instead of general inquiries.

3.3.4 PERFORMANCE OF OPEN SET UOT

To further assess the practicality of UoT, we test it in the open set setting and compare its performance with DPOS. As shown in Table 3, UoT enhances DPOS by an average relative improvement of 17.4%.

3.3.5 FURTHER ANALYSIS

Our study finds that the UoT’s one-step planning performs well, thanks to effective reward design and question selection. Although deeper planning improves performance, we cap simulations at three steps for budget reasons, striking a balance between efficiency and effectiveness. We evaluated GPT-4’s accuracy across datasets, analyzing 10% of interactions. GPT-4 showed consistent accuracy and reliability. See Appendix 6.3 and 6.4 for details and results.

4 RELATED WORK

Planning and Reasoning of LLMs LLMs show prowess in planning and reasoning. Wei et al. (2022) introduced CoT prompting for intermediate reasoning; Yao et al. (2023) proposed ToT prompting using DFS/BFS. Besta et al. (2023) present GoT to solve elaborate problems. Feng et al. (2023) illustrated TS-LLM’s tree-search guided decoding. ReAct Yao et al. (2022) offers acting-based prompting, while Reflexion Shinn et al. (2023) enhances this with feedback reflection. Zhou et al. (2023) unify reasoning and planning.

Decision-making and Information-seeking by LLMs LLMs have evolved as decision-making tools, with models like LLM+P Liu et al. (2023a) and LLM-DP Dagan et al. (2023) combining external planners and LLMs for natural language-based programming. RAP Hao et al. (2023) goes beyond structured language, using LLMs with Monte Carlo Tree Search (MCTS) Chaslot et al. (2008) for dynamic decision-making. This approach is also seen in the work of Zhao et al. (2023), applying MCTS and LLM knowledge for complex tasks like robot control. However, MCTS struggles in uncertain scenarios due to its reliance on terminal states and specific modules for rewards and action selection. Additionally, to enhance LLMs’ questioning abilities, Deng et al. (2023) introduce the Rephrase and Respond method. AVIS Hu et al. (2023) represents an autonomous visual question answering system that uses external tools. Pan et al. (2023) introduce KwaiAgents for processing queries, following guidelines, and accessing external documents.

5 CONCLUSION AND DISCUSSION

This paper presents the Uncertainty of Thoughts (UoT) algorithm, significantly improving LLMs in tasks requiring active information seeking through tree-based simulation, uncertainty-based rewards and a reward propagation scheme. On five datasets UoT increases success rate by 57.8% on average, establishing a new benchmark for evaluating LLMs in active information-seeking tasks. We evaluate UoT on simplified scenarios; more realistic scenarios raise challenges like allowing incomplete elimination of possibilities by answers, and issues with completely open-ended questions/answers, which we leave for future work. We further discuss these limitations and future work in Appendix 6.6.

REFERENCES

- Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernandez Abrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vlad Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, Guy Gur-Ari, Steven Hand, Hadi Hashemi, Le Hou, Joshua Howland, Andrea Hu, Jeffrey Hui, Jeremy Hurwitz, Michael Isard, Abe Ittycheriah, Matthew Jagielski, Wenhao Jia, Kathleen Kenealy, Maxim Krikun, Sneha Kudugunta, Chang Lan, Katherine Lee, Benjamin Lee, Eric Li, Music Li, Wei Li, YaGuang Li, Jian Li, Hyeontaek Lim, Hanzhao Lin, Zhongtao Liu, Frederick Liu, Marcello Maggioni, Aroma Mahendru, Joshua Maynez, Vedant Misra, Maysam Moussalem, Zachary Nado, John Nham, Eric Ni, Andrew Nystrom, Alicia Parrish, Marie Pellat, Martin Polacek, Alex Polozov, Reiner Pope, Siyuan Qiao, Emily Reif, Bryan Richter, Parker Riley, Alex Castro Ros, Aurko Roy, Brennan Saeta, Rajkumar Samuel, Renee Shelby, Ambrose Slone, Daniel Smilkov, David R. So, Daniel Sohn, Simon Tokumine, Dasha Valter, Vijay Vasudevan, Kiran Vodrahalli, Xuezhi Wang, Pidong Wang, Zirui Wang, Tao Wang, John Wieting, Yuhuai Wu, Kelvin Xu, Yunhan Xu, Linting Xue, Pengcheng Yin, Jiahui Yu, Qiao Zhang, Steven Zheng, Ce Zheng, Weikang Zhou, Denny Zhou, Slav Petrov, and Yonghui Wu. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.
- Anthropic. Claude 2, 2023. URL <https://www.anthropic.com/index/claude-2>.
- Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michal Podstawski, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. *arXiv preprint arXiv:2308.09687*, 2023.
- Guillaume Chaslot, Sander Bakkes, Istvan Szita, and Pieter Spronck. Monte-carlo tree search: A new framework for game ai. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 4, pp. 216–217, 2008.
- Cheng-Han Chiang and Hung-yi Lee. Can large language models be an alternative to human evaluations? *arXiv preprint arXiv:2305.01937*, 2023.
- Cohere. Cohere for ai, 2023. URL <https://cohere.com/>.
- Gautier Dagan, Frank Keller, and Alex Lascarides. Dynamic planning with a llm. *arXiv preprint arXiv:2308.06391*, 2023.
- Yihe Deng, Weitong Zhang, Zixiang Chen, and Quanquan Gu. Rephrase and respond: Let large language models ask better questions for themselves. *arXiv preprint arXiv:2311.04205*, 2023.
- Xidong Feng, Ziyu Wan, Muning Wen, Ying Wen, Weinan Zhang, and Jun Wang. Alphazero-like tree-search can guide large language model decoding and training. *arXiv preprint arXiv:2309.17179*, 2023.
- Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*, 2023.
- Ziniu Hu, Ahmet Iscen, Chen Sun, Kai-Wei Chang, Yizhou Sun, David A Ross, Cordelia Schmid, and Alireza Fathi. Avis: Autonomous visual information seeking with large language model agent. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.
- Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*, 2023a.

- Wenge Liu, Jianheng Tang, Yi Cheng, Wenjie Li, Yefeng Zheng, and Xiaodan Liang. Meddg: an entity-centric medical consultation dataset for entity-aware medical dialogue generation. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pp. 447–459. Springer, 2022.
- Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: Nlg evaluation using gpt-4 with better human alignment, may 2023. *arXiv preprint arXiv:2303.16634*, 2023b.
- David Noever and Forrest McKee. Chatbots as problem solvers: Playing twenty questions with role reversals. *arXiv preprint arXiv:2301.01743*, 2023.
- OpenAI. Gpt-3.5 turbo: A high-performance language model, 2023a. URL <https://www.openai.com/research/gpt-3-5-turbo>. Whitepaper.
- OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023b.
- Haojie Pan, Zepeng Zhai, Hao Yuan, Yaojia Lv, Ruiji Fu, Ming Liu, Zhongyuan Wang, and Bing Qin. Kwaiaagents: Generalized information-seeking agent system with large language models. *arXiv preprint arXiv:2312.04889*, 2023.
- Joon Sung Park, Joseph O’Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pp. 1–22, 2023.
- Dinesh Raghu, Shantanu Agarwal, Sachindra Joshi, and Mausam. End-to-end learning of flowchart grounded task-oriented dialogs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 4348–4366, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.357. URL <https://aclanthology.org/2021.emnlp-main.357>.
- Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.
- Noah Shinn, Beck Labash, and Ashwin Gopinath. Reflexion: an autonomous agent with dynamic memory and self-reflection. *arXiv preprint arXiv:2303.11366*, 2023.
- Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adria Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. *arXiv preprint arXiv:2305.04091*, 2023.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.
- Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*, 2023.

Lin Xu, Qixian Zhou, Ke Gong, Xiaodan Liang, Jianheng Tang, and Liang Lin. End-to-end knowledge-routed relational dialogue system for automatic diagnosis. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 7346–7353, 2019.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.

Zirui Zhao, Wee Sun Lee, and David Hsu. Large language models as commonsense knowledge for large-scale task planning. *arXiv preprint arXiv:2305.14078*, 2023.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*, 2023.

6 APPENDIX

6.1 DERIVATION OF INFORMATION GAIN FORMULA

Recall that the information gain at node v is defined as the expected change in uncertainty (or entropy) when receiving an answer at this node, which we defined as:

$$IG_v(X) := H_v(X) - p_v^Y \cdot H_v^Y(X) - p_v^N \cdot H_v^N(X) \quad (12)$$

We now show that:

Proposition 1. *The information gain at node v is equal to:*

$$IG_v(X) = -p_v^A \log p_v^A - p_v^N \log p_v^N \quad (13)$$

Proof. Note that for any outcome x_i , we have by the rules of conditional probability:

$$p(x_i|\Omega_v^A) = \frac{p(x_i|\Omega_v)}{p(\Omega_v^A|\Omega_v)} = \frac{p(x_i|\Omega_v)}{p_v^A} \quad (14)$$

Now the information gain is:

$$\begin{aligned} IG_v(X) &= H_v(X) - p_v^A \cdot H_v^A(X) - p_v^N \cdot H_v^N(X) \\ &= - \sum_{i:\omega_i \in \Omega_v} p(x_i|\Omega_v) \log p(x_i|\Omega_v) \\ &\quad + p_v^A \sum_{i:\omega_i \in \Omega_v^A} p(x_i|\Omega_v^A) \log p(x_i|\Omega_v^A) \\ &\quad + p_v^N \sum_{i:\omega_i \in \Omega_v^N} p(x_i|\Omega_v^N) \log p(x_i|\Omega_v^N) \\ &= \sum_{i:\omega_i \in \Omega_v^A} p(x_i|\Omega_v^A) (\log p(x_i|\Omega_v^A) - \log p(x_i|\Omega_v)) \\ &\quad + \sum_{i:\omega_i \in \Omega_v^N} p(x_i|\Omega_v^N) (\log p(x_i|\Omega_v^N) - \log p(x_i|\Omega_v)), \end{aligned}$$

where the last equality holds by $p_v^A \cdot p(x_i|\Omega_v^A) = p(x_i|\Omega_v)$, and similarly for p_v^N . We further compute that

$$\begin{aligned} &\sum_{i:\omega_i \in \Omega_v^A} p(x_i|\Omega_v^A) (\log p(x_i|\Omega_v^A) - \log p(x_i|\Omega_v)) \\ &= \sum_{i:\omega_i \in \Omega_v^A} p(x_i|\Omega_v^A) \log \frac{p(x_i|\Omega_v^A)}{p(x_i|\Omega_v)} \\ &= - \sum_{i:\omega_i \in \Omega_v^A} p(x_i|\Omega_v^A) \log p_v^A \\ &= -p_v^A \log p_v^A \end{aligned}$$

Analogously the remaining term is $-p_v^N \log p_v^N$. Finally we conclude that

$$IG_v(X) = -p_v^Y \log p_v^Y - p_v^N \log p_v^N \quad (15)$$

□

In fact, this proposition can also be proven using some properties of information theory, particularly the definitions of conditional entropy and mutual information. As the more computational proof shown here is still relatively short and does not require defining certain additional probability distributions, we provide the computational proof here instead.

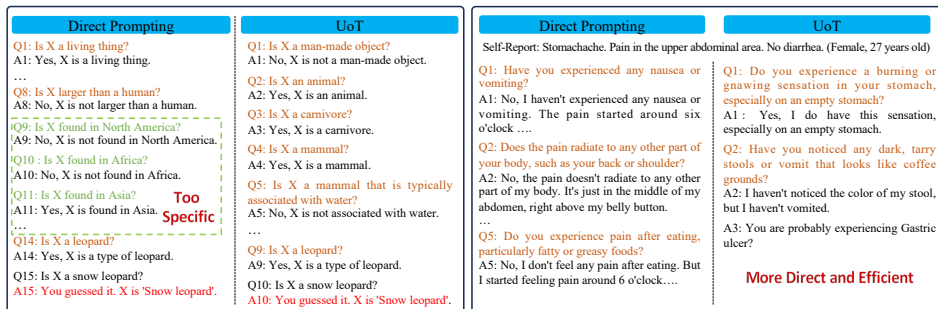


Figure 4: Case studies from the 20 Questions game (left) and simplified medical diagnosis (right).

6.2 CASE STUDY EXAMPLE

As shown in Figure 4, these two examples can further demonstrate the effectiveness of our UoT method.

6.3 EFFECT OF SIMULATION DEPTH

As Figure 5 illustrates, we analyze the impact of simulation steps. Even with one-step reasoning and planning, our method can still have a strong performance, further indicating the effectiveness of our reward design and question selection mechanism. With the increase of the step, the performance can gradually rise. However, due to the constraints of computation resources and OpenAI API budgets, we only explore the simulation to the third step and argue that it can be the practical tradeoff between performance and efficiency.

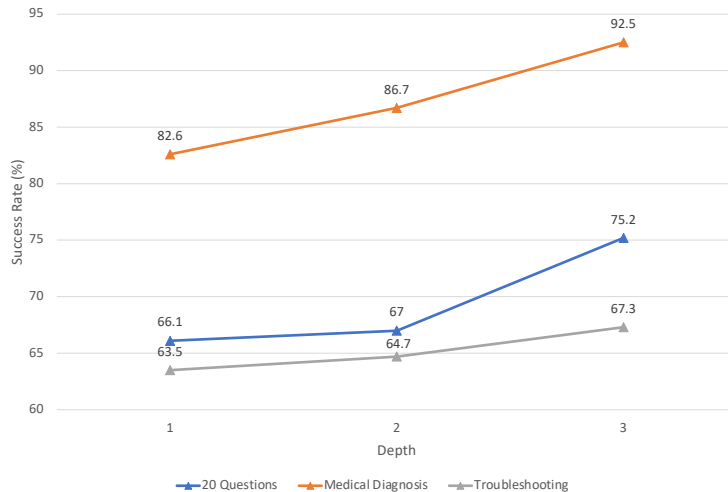


Figure 5: Results comparison of different depth settings in UoT.

6.4 RELIABILITY OF GPT-4 AS THE ENVIRONMENT

As the impressive understanding ability of LLMs, previous research has validated the effectiveness of evaluators served by ChatGPT or GPT-4 Chiang & Lee (2023); Liu et al. (2023b). Consequently, we also adopt GPT-4 as the environment to provide feedback on our work. Prompts can be found in Appendix 6.9.4. To assess the accuracy and reliability of employing GPT-4 as the environment simulator, we randomly sample 10% interaction records (including the final judgment and intermediate feedback from the environment) from each dataset. As Figure 4 shows, GPT-4 can provide completely accurate judgment and also keep a high level of accurate feedback during the interaction. These experimental results can further support the effectiveness of our method.

Table 4: Human evaluation results for the accuracy of environment feedback served by GPT-4. IF represent the Accuracy of **I**ntermediate **F**eedback

Scenario	Judgement	IF
20 Questions	100	93.7
Medical Diagnosis	100	94.4
Troubleshooting	100	92.9

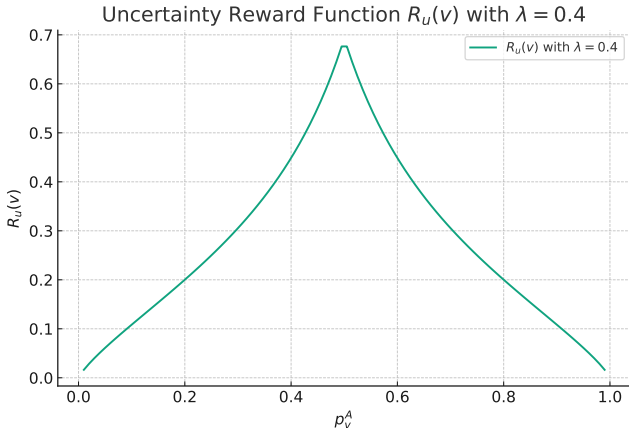


Figure 6: Curve of uncertainty based reward on Eq 10, where p_v^N can be replaced by $(1 - p_v^A)$. The horizontal axis p_v^A is conditional probabilities of affirmative at node v , which are introduced in Section §2.4.

6.5 REWARD FUNCTION DETAILS AND ITS CURVE

Refer to Figure 6 for the curve of uncertainty-based reward function.

6.6 LIMITATION AND FUTURE WORK

In practice, Ω_v^A and Ω_v^N might overlap, as different answers (such as “yes” or “no”) may lead to the exclusion of different sets of possibilities. Another similar limitation is that some questions or answers may not fully eliminate certain possibilities (e.g., “I don’t have a fever” does not 100% eliminate the possibility of having COVID-19). Furthermore, compared to completely open-ended interaction in medical diagnosis or troubleshooting, our current benchmark represents a simplified scenario. In theory, such cases could be handled using the method of converting interactions into probability estimations and applying some kind of Bayesian update to the probabilities of each possibility, rather than just eliminating some subset.

6.7 EXPERIMENTAL SETUPS

6.7.1 BASELINES SETUP

Chain-of-Thought (CoT) We adapt the typical CoT prompt which instruct LLM to generate the explanation or motivation for the proposed question first, then give the question to ask.

Chain of Thought with Self-Consistency (CoT-SC) To make the method spend comparable compute to our approach for a fair comparison, we sampled 33 times before deciding on each action with the LLM’s temperature of 0.7. The final selected question is the one repeated most times among 33 samples.

Planning Prompting To measure whether LLMs’ planning ability can be enhanced through some crafted prompts like CoT, ToT or Reflexion. We design the prompt to enable LLM to simulate

multiple different sets of future interactions between questioner and answerer, then let LLM choose one most promising interaction (question) to ask.

Tree of Thoughts In the case of **Original-ToT**, a sampling method is employed to generate 3 questions from each answer node, and the self-evaluation method is utilized for reward calculation. Subsequently, breadth-first search will be used and 10 nodes from each step will be selected for later simulation. Additionally, the temperature of the LLM is configured to 0.7, consistent with the settings in original ToT paper. In the case of **Adapted-ToT**, we provide more heuristical hints in prompt to generate the questions, e.g. ‘you should try to propose the question to halve the probability set. Likewise, each answer node generates 3 questions, and the LLM selects 10 nodes with higher self-evaluation rewards to further simulation. The simulation steps are also 3.

Reflexion This approach involves the LLM agent suggesting questions iteratively until the question reward exceeds the threshold of 0.7 or reaches the maximum limit of 3 questions. The reward score s is calculated using the formula $s = \min(p^A, p^N) / \max(p^A, p^N)$. This heuristic is based on the principle of whether the question can effectively halve the probability set. If a candidate question achieves a score above the threshold, the process of proposing questions is concluded, and that question is selected. In cases where no question meets the threshold, the one with the highest score is chosen.

Uncertainty of Thoughts Pruned After generating the candidate question based on the possibility set Ω_i , we sorted these question nodes by uncertainty based reward and reserved half of them, serving the purpose of pruning. In subsequent steps of the simulation, this pruning operation will be continued. Other settings were the same as UoT, described in Section §6.7.3.

6.7.2 SCENARIOS SETTINGS AND DATASETS

20 Questions game is a classic guessing game where the *answerer* thinks of an object, person, place, or other, and the *questioner*, possessing no prior knowledge about the chosen entity, proceeds to pose a series of up to 20 yes-or-no questions to determine what the secret item is. The questions are designed to narrow the possibilities and ultimately guess the secret item within the 20 questions. **20 Questions in BIG-bench**: It is the sub-task of BIG-bench and can be found on the GitHub website⁶. **Common Dataset Construction**: We came across an official website⁷ that introduces a 20 Questions game, which mentions that common target categories in this game include animals, places, food, and objects. Therefore, we extracted and manually screened the targets mentioned on this website, resulting in a dataset named ‘‘Common’’ comprising 111 targets, each belonging to one of the four aforementioned categories.

Medical Diagnosis In this scenario, the patient will simply describe their symptom first which we call a ‘Self-report’, then doctor acted by LLM will start to ask questions to interact with patient to determine the disease.

Troubleshooting In FloDial dataset, it includes faults of car and laptop. Similar to Medical Diagnosis, the customer first describes some simple fault symptoms, then the customer support technician will chat with customer to further check the specific issues of device.

LLMs Serve as Questioner (Patient or Customer) In simulated interactions involving questioner and answerer scenarios, particularly for medical diagnosis and troubleshooting, the response given by an LMM acting answerer is guided by scenario instructions and real-world dialogue examples. This approach makes the LM’s responses more human-like and enhances its accuracy in diagnosing diseases or identifying faults. While, in a game like ‘‘20 Questions,’’ where the objective is to guess common items, the LMM acting as the answerer only needs to provide simple ‘yes’ or ‘no’ answers. Therefore, incorporating real-world dialogue into the LMM’s prompts for this game is not necessary.

6.7.3 IMPLEMENTATION

Empirically, we set the plan (simulation) steps as 3 and the number of questions during the simulation is 3. The hyperparameter λ in uncertainty-based reward is 0.4.

⁶https://github.com/google/BIG-bench/tree/main/bigbench/benchmark_tasks/twenty_questions

⁷<https://blog.prepscholar.com/20-questions-game>

6.8 EXAMPLES IN SCENARIOS

Some examples of conversation in different scenarios are provided in Table 5-9.

Table 5: Example for 20 Question. **Q** stands for Questioner and **A** stands for Answerer.

Target: Glass micropipette	
Conversation	
1: Q: Is X a tangible object?	A: Yes.
2: Q: Is X a living organism?	A: No.
3: Q: Is X a household item?	A: No.
4: Q: Is X a type of transportation?	A: No.
5: Q: Is X used in construction?	A: No.
6: Q: Is X a wearable item?	A: No.
7: Q: Is X a type of toy?	A: No.
8: Q: Is X a 'Glass micropipette'?	
A: You guessed it. X is a 'Glass micropipette'.	

Table 6: Example for Medical Diagnosis. **D** stands for Doctor (Questioner) and **P** stands for Patient (P).

Disease: Allergic rhinitis
Self-report
Hello doctor, my son's throat easily becomes red and swollen as soon as winter comes. He has been coughing and his throat has been red and swollen for the past half month.
Conversation
1: D: Do you experience a runny or stuffy nose? P: Yes.
2: D: You may have 'Allergic rhinitis'.

Table 7: Example for MedDG. **D** stands for Doctor (Questioner) and **P** stands for Patient (Answerer).

Disease: Gastritis
Self-report
Recently, I always feel nauseous when I wake up in the morning, but during the day, everything seems fine. (Female, 33 years old)
Conversation
1: D: Have you been experiencing vomiting or diarrhea recently? P: No, I haven't. I just feel nauseous and my stomach is uncomfortable.
2: D: Have you noticed any dark, tarry stools or vomit that looks like coffee grounds? E: No, I haven't noticed anything like that.
3: D: Are you experiencing pregnancy-related morning sickness? P: No, I'm not pregnant. It's just that my stomach feels uncomfortable and nauseous.
4: D: You may have 'Gastritis'.

6.9 PROMPTS

6.9.1 DPOS AND DPCS PROMPT

The specific prompts for the DPOS and DPCS baselines in each scenario are provided in Table 10-12.

6.9.2 PLANNING PROMPT

The specific prompts for Planning Prompt baselines in each scenario are provided in Table 13, Table 14 and Table 15. As planning prompt method is close set setting, hence the probability set will also be informed in the prompt as DPCS prompt. We do not repeat it in the tables.

Table 8: Example for MedDG. **D** stands for Doctor (Questioner) and **P** stands for Patient (Answerer).

Disease: Gastric ulcer
Self-report Stomachache. Pain in the upper abdominal area. No diarrhea. (Female, 27 years old)
Conversation
1: D: Do you experience a burning or gnawing sensation in your stomach, especially on an empty stomach?
P: Yes, I do have this sensation, especially on an empty stomach.
2: D: Have you noticed any dark, tarry stools or vomit that looks like coffee grounds?
P: I haven't noticed the color of my stool, but I haven't vomited.
3: D: You may have 'Gastric ulcer'.

Table 9: Example for Troubleshooting. **T** stands for customer support technician and **C** stands for customer.

Fault: Brake fluid overflow
Self-report Lately, I have been having trouble with the brakes of my car. I have checked it again and again but can't figure out what's causing this. Please help me.
Conversation
2: T: Does the issue affect the rotation or movement of the wheels? C: No.
3: T: Does the issue seem to be related to the overall braking system? C: Yes.
4: T: Are you experiencing a brake fluid leak? C: No.
5: T: Are your brake pads worn out? C: No.
6: T: Is the brake pedal feeling spongy or soft when you press it? C: Yes.
7: T: Is there an issue with the brake lines being air-filled or damaged? C: No.
8: T: Your car may have a brake fluid overflow issue.

6.9.3 UOT PROMPT

The detailed prompts for our UoT method in each scenario are attached in Table 16, Table 17 and Table 18.

6.9.4 QUESTIONER PROMPT

The specific prompt to guide the examiner in all scenarios is provided in Table 19, Table 20 and Table 21.

Table 10: DPOS and DPCS Prompt for 20 Question game. The two settings are the same except for the content noted in red, which is only for DPCS Prompt. The first time the content marked in red is mentioned at the beginning of interaction is to inform the questioner of the entire probability set. The second mention serves as a reminder for the questioner to determine the target as soon as possible when the conversation is nearing its end. We establish two phases with the aim of encouraging the questioner to guess the target as soon as possible when the second phase of interaction is nearing its conclusion, in order to avoid failure at the end of interaction.

Prologue
Let us play the game of 20 questions. I am impersonating the thing, X. You will ask me up to 20 questions that start with 'Is X' and can only be answered by yes or no, and I will answer each one truthfully based on being X.
Let us begin. Ask me the first question. Reply to me with your first question.
The X may be one of the items below: [list of items]
Phase 1 (Q1 - Q13)
<i>None</i>
Phase 2 (Q14 - Q20)
The X may be one of the items below: [list of items]

Table 11: DPOS and DPCS are medical diagnostic prompts with similar structures. The key difference is the inclusion of probability set information in red text for DPCS. This red text appears twice: initially to inform the questioner about all potential diagnoses and again towards the end of the interaction as a reminder to quickly confirm the disease. We establish two phases with the aim of encouraging the questioner to confirm the disease as soon as possible when the second phase of interaction is nearing its conclusion, avoiding failure at the end of interaction.

Prologue
You are a doctor and your patient reports that: [self-report]
You should ask your patient questions in English with symptoms which can only be answered by yes or no, to find what disease this patient suffers.
Let us begin. Ask me the first question.
Based on the symptoms above, if you find out the disease, please ask 'Are you a [disease name]?'
The patient may suffer from one of the diseases below: [list of diseases]
Phase 1 (Q1)
<i>None</i>
Phase 2 (Q3 - Q5)
Based on the symptoms above, if you find out the disease, please ask 'Are you a [disease name]?'
The patient may suffer from one of the diseases below: [list of diseases]

Table 12: DPOS and DPCS are troubleshooting prompts with similar structures, but DPCS includes unique content highlighted in red. This red content appears first at the beginning, outlining all potential faults, and again towards the end as a reminder to swiftly identify the fault. The two-phase structure of these prompts aims to ensure quick fault confirmation, especially in the final stages of the interaction, to prevent failure.

Prologue
You are a technician and your client reports that: [self-report]
You should ask your client questions about a specific situation which can only be answered by yes or no, in order to find where the issue this client faces with located.
Let us begin. Ask me the first question.
The client may face one of the issues below: [list of issues]
Phase 1 (Q1 - Q13)
<i>None</i>
Phase 2 (Q14 - Q20)
Based on the situations above, if you find out the issue, please ask 'Are you a [issue name]?'
The client may face one of the issues below: [list of issues]

Table 13: Planning Prompt for 20 Question game. [C1] is the count of questions asked and [C2] is the count of questions remaining. The ‘information gained’ marked blue represents the previous interaction history. We divide it into three phases to discuss the probability set as quickly as possible, conduct simulation for planning, and remind the questioner to guess the answer.

Prologue
<i>Same as prompts in Appendix 6.9.1</i>
Phase 1 (Q1 - Q4)
The next question should narrow down the possible range of X, preferably in half.
Phase 2 (Q5 - Q15)
We are playing the 20 Question game, [C1] questions have been asked. And now we know: [information gained]
Based on the features of X above, please guess what X exactly is and tell me your top 3 most likely answers.
For these three candidate X, please separately complete the remaining [C2] questions and answer yes/no by yourself. Notably, you must guess the corresponding X before the last question.
Phase 3 (Q16 - Q20)
Note that you should guess what X exactly is from now on. The question must start with ‘Is X ...’

Table 14: Planning Prompt for Medical Diagnosis. [C1] is the count of questions asked and [C2] is the count of questions remaining. The ‘information gained’ marked blue represents the previous interaction history. We divide it into three phases to discuss the probability set as quickly as possible, conduct simulation for planning, and remind the questioner to confirm the disease.

Prologue
<i>Same as prompts in Appendix 6.9.1</i>
Phase 1
<i>Skip because of the limited QA rounds in this scenario</i>
Phase 2 (Q1 - Q3)
You are the doctor asking questions to diagnose, [C1] questions have been asked. And now we know about the patient: [information gained]
Based on the symptoms of the patient above, please think about what disease the patient suffers from and tell me your top three most likely answers.
For these three candidate diseases, please separately complete the remaining [C2] questions and answer yes/no by yourself. Notably, you must determine the corresponding disease before the last question.
Phase 3 (Q4 - Q5)
Note that you should determine what disease the patient suffers from now. The question must start with ‘Are you a [disease name]?’

Table 15: Planning Prompt for Troubleshooting. [C1] is the count of questions asked and [C2] is the count of questions remaining. The ‘information gained’ marked blue represents the previous interaction history. We divide it into three phases to discuss the probability set as quickly as possible, conduct simulation for planning, and remind the questioner to confirm the fault.

Prologue
<i>Same as prompts in Appendix 6.9.1</i>
Phase 1 (Q1 - Q4)
The next question should narrow down the possible range of trouble issues, preferably in half
Phase 2 (Q5 - Q15)
You are a technician to troubleshoot, [C1] questions have been asked. And now we know: [information gained]
Based on the situation your client faces, please think about what the issue exactly is and tell me your top 3 most likely answers.
For these three candidate issues, please separately complete the remaining [C2] questions and answer yes/no by yourself. Notably, you must determine the corresponding issue before the last question.
Phase 3 (Q16 - Q20)
Note that you should determine what issue your client faces from now on. The question must start with ‘Are you a [issue name]?’

Table 16: UoT Prompt for the 20 Questions Game: As it is based on a closed-set setting, information about probabilities will be given at the beginning of the interaction and will be reminded after Q14. Since it is similar to previous prompts, we will not repeat it here. In the ‘Prompt for Question Generation and Simulation’, the count of YES/NO indicates the number of items that are consistent with the affirmative/negative response.

Prologue
Let us play the game of 20 questions. I am impersonating the thing, X. You will ask me up to 20 questions that start with ‘Is X’ and can only be answered by yes or no, and I will answer each one truthfully based on being X.
Let us begin. Ask me the first question. Reply to me with your first question.
Prompt for Question Generation and Simulation
Please design a question about X and can only be answered by YES or NO. asked Then classify the possible X above based on this question. If the answer is ‘YES’, put this X into ‘YES: ...’, otherwise to ‘NO: ...’. Finally calculate how many X in YES and NO.
Notably, this question should fulfill that the count of YES and NO are almost the same with a permissible discrepancy of no more than one!
You should think about best n questions to respond to. And your answer should be:
Question 1: Is X ...?
YES: item1, item2, ...
Count of YES: ...
NO: item1, item2, ...
Count of NO: ...
Additional Reminder in Q14 - Q20
Note that you should guess and ask what X exactly is from now on. X is possible a: [item_list_str], or other. The question must start with ‘Is X ...

Table 17: UoT Prompt for medical diagnosis: As it is based on a closed-set setting, information about probabilities will be given at the beginning of the interaction and will be reminded after Q3. Since it is similar to previous prompts, we will not repeat it here. In the 'Prompt for Question Generation and Simulation', the count of YES/NO indicates the number of diseases that are consistent with the affirmative/negative response.

Prologue

You are a doctor and your patient reports that: [\[self-report\]](#)

You should ask your patient questions in English with symptoms which can only be answered by yes or no, to find what disease this patient suffers.

Let us begin. Ask me the first question.

Prompt for Question Generation and Simulation

Please design a question to ask your patient with symptoms about disease and can only be answered by YES or NO. Then classify the possible disease above based on each question. If the answer is 'YES', put this disease into 'YES: ...', otherwise to 'NO: ...'. Finally calculate how many X in YES and NO.

Notably, this question should fulfill that the count of YES and NO are almost the same with a permissible discrepancy of no more than one!

You should think about best n questions to respond to.

And your answer should be:

Question 1: ...?

YES: disease1, disease2, ... (disease names only)

Count of YES: ...

NO: disease1, disease2, ... (disease names only)

Count of NO: ...

Additional Reminder in Q3 - Q5

Note that you should point out and ask what disease the patient suffers from now. The patient may suffer from one of diseases below: [list of disease], or other. The question must be 'You may have a [disease name]?'

Table 18: UoT Prompt for troubleshooting: As it is based on a closed-set setting, information about probabilities will be given at the beginning of the interaction and will be reminded after Q14. Since it is similar to previous prompts, we will not repeat it here. In the 'Prompt for Question Generation and Simulation', the count of YES/NO indicates the number of faults which are consistent with the affirmative/negative response.

Prologue

You are a technician and your client reports that: [\[self-report\]](#)
 You should ask your client questions about a specific situation which can only be answered by yes or no, in order to find where the issue this client faces with located.
 Let us begin. Ask me the first question.

Prompt for Question Generation and Simulation

Please design a question to ask your client with specific situation and can only be answered by YES or NO. asked Then classify the possible issue above based on this question. If the answer is 'YES', put this issue into 'YES: ...', otherwise to 'NO: ...'. Finally calculate how many issues in YES and NO.

Notably, this question should fulfill that the count of YES and NO are almost the same with a permissible discrepancy of no more than one!

You should think about best n questions to respond to. And your answer should be:
 Question 1: ...?
 YES: issue1, issue2, ... (issue names only)
 Count of YES: ...
 NO: issue1, issue2, ... (issue names only)
 Count of NO: ...

Additional Reminder in Q14 - Q20

Note that you should point out and ask what disease the patient suffers from now. The patient may suffer from one of diseases below: [list of disease], or other. The question must be 'You may have a [disease name]?'

Table 19: Prompt for Answerer in 20 questions game. [\[target\]](#) is the name of the final target for each dialogue.

Prologue

Let us play the game of 20 questions. You are the answerer and I am the guesser. X is '[\[target\]](#)'. I will ask you up to 20 questions and you should answer each one truthfully based on being X. If I guess correctly what is X, answer me "You guessed it. X is '[\[target\]](#)'." Note that never directly tell me what X is all the time.
 Let us begin. Here is my first question.

Table 20: Prompt for Answerer in medical diagnosis. [\[disease\]](#) is the name of the final disease for each dialogue.

Prologue

You are the patient suffering '[\[target\]](#)' and I am the doctor. I will ask you up to 5 questions and you should answer each one truthfully based on your disease. If I point out correctly what disease you experience, answer me "You are right. I am experiencing '[\[target\]](#)'." Note that never directly tell me what disease is all the time.
 Let us begin. Here is my first question.

Table 21: Answerer Prompt in troubleshooting. [\[fault\]](#) is the name of the final fault for each dialogue.

Prologue

You are the client with a device that has '[\[target\]](#)' and I am the technician. I will ask you up to 20 questions and you should answer each one truthfully based on the issue of your device. If I point out correctly what your issue is, answer me "You are right. My device has '[\[target\]](#)'." Note that never directly tell me what the issue is all the time.
 Let us begin. Here is my first question.
