GoT: A Game-Theoretic Approach to the Game of Twenty Questions

Langyuan Cui, Hwee Tou Ng,* Chun Kai Ling*
Department of Computer Science
National University of Singapore
13 Computing Drive, Singapore 117417
langyuan.c@u.nus.edu, {dcsnght, chunkail}@nus.edu.sg

Abstract

The *Game of Twenty Questions* is a classical problem where a questioner must identify a hidden item by asking binary questions. Although simple in form, the game is strategically challenging, especially when the questions are expressed in natural language. It is also a useful abstraction for realistic information-seeking tasks, such as medical diagnosis and troubleshooting. Existing approaches often rely on simplifying assumptions that degrade *worst-case* performance. This is an issue with serious implications in high-stakes applications. In this work, we introduce *Strategic Language Search* (SLS), an adversarial counterpart of Twenty Questions, and formalize it and its variants as two-player zero-sum extensive-form games. We then propose *Game of Thought* (GoT), a framework that applies game-theoretic techniques to approximate a Nash equilibrium (NE) strategy for the restricted variant of the game. Empirical results demonstrate that our approach consistently improves worst-case performance compared to (1) direct prompting-based methods and (2) heuristic-guided search methods across all tested settings.

1 Introduction

The *Game of Twenty Questions* is a classical game of sequential information acquisition [41]. An item is selected from a known set, and the questioner must identify it by asking a sequence of Yes/No questions. Despite its simple rules, the game poses a challenging strategic problem: each question must be chosen based not only on its immediate usefulness, but also on how it shapes the remaining space of possible items and future questions. This is further amplified when questions are expressed in natural language, since the space of all possible natural language questions is too large to exhaustively explore. As a result, fully solving the game is generally infeasible.

Twenty Questions is closely related to the broader problem of information seeking. In settings such as planning [51], medical diagnosis [32], and other partially observable tasks [30], an agent may need to acquire missing information through interaction, often by asking clarification questions. We view Twenty Questions as a structured abstraction of this broader problem. It captures the core challenge of sequentially selecting informative queries, while the restriction to binary questions provides sufficient structure needed to define, solve, and evaluate methods in a principled manner. This restriction does not make the problem trivial, and as [52, 20] show, naive approaches perform poorly, highlighting the strategic complexity of Twenty Questions. Furthermore, the binary setting remains practically relevant: domains such as medical diagnosis and troubleshooting naturally involve sequences of yes/no questions or diagnostic tests. This makes Twenty Questions a simple yet meaningful setting.

Prior works have studied questioning policies using LLM-based reasoning methods, such as Self-Consistency (SC) [47] and Tree of Thoughts (ToT) [50]. More recently, [20] proposed Uncertainty

*These authors jointly supervised the first author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

of Thoughts (UoT), which uses tree search to *explicitly model and optimize* the information gained from questions. However, UoT makes the strong assumption that the target item is chosen uniformly at random. Often, assuming a known distribution over items is inappropriate, either because the underlying context may not naturally admit a probabilistic interpretation [21], or because such distributions are non-uniform and difficult to estimate.

Motivated by this, we study Twenty Questions from a worst-case rather than average-case perspective. This objective is relevant in both adversarial and non-adversarial settings. For instance, in cybersecurity incident response, an attacker may strategically choose which component to compromise, making policies based on historical attack frequencies potentially vulnerable. In non-adversarial settings such as medical diagnosis and troubleshooting, worst-case optimization is also meaningful when the distribution over underlying cases is unknown or unreliable. A mechanic diagnosing vehicle faults, for example, may not have access to reliable fault frequencies for a particular vehicle model. Such situations motivate distribution-free strategies that avoid relying on potentially inaccurate priors, consistent with the broader literature on minimax decision making and decision making under uncertainty [46, 13, 18]. We therefore propose *Game of Thought* (GoT), a game-theoretic framework that models Twenty Questions as a two-player, zero-sum extensive-form game with imperfect information. The questioner seeks to identify the item efficiently, while the adversary chooses items that maximize the number of questions required. By approximating the Nash equilibrium (NE) of this game, GoT obtains questioning strategies that optimize for the worst-case performance, making it robust to *all* distributions. We emphasize that GoT does not assume the setting is adversarial in practice. This formulation is a game-theoretic device for optimizing worst-case performance without the need for strong assumptions about item distributions, making GoT particularly suitable for settings where reliable priors are difficult to obtain.

Contributions We **(i)** formulate the Strategic Language Search (SLS) problem and its variants, **(ii)** analyze the game-theoretic structure of SLS, including the need for randomized strategies under adversarial item selection, **(iii)** propose GoT, a novel game-theoretic framework to approximate the NE in the SLS-restricted variant, and **(iv)** empirically demonstrate that in the game of Twenty Questions and relevant practical settings including medical diagnosis and troubleshooting, GoT is consistently superior to (a) pure prompting-based and (b) search-based methods.

2 Related Work

Uncertainty of Thoughts GoT is most closely related to *Uncertainty of Thoughts* (UoT) [20], which studies question selection in the Game of Twenty Questions. UoT performs depth-limited exhaustive search over candidate questions to maximize expected information gain, assuming that the target item is chosen uniformly at random. This makes UoT a strong search baseline, and an upper-bound-like comparison in terms of average performance to sampling-based methods such as Monte Carlo Tree Search (MCTS) [7]. However, UoT optimizes an average-case objective under a *known, fixed* distribution. In contrast, GoT formulates the task as adversarial and optimizes for worst-case performance, avoiding the need for a known item distribution.

Natural Language in Games We study games in which the *action space* is implicitly given by natural language questions. This differs from much of the prior work on natural language in games, where language is used as a communication channel between players, for instance to negotiate [24], share information [40], or coordinate [1]. Our setting resembles signaling games [29, 10, 14], although classical signaling games typically assume a finite set of signals. Natural language, by contrast, gives rise to an intractably large action space. Prior works addressed this using techniques such as sentiment analysis [5, 19] or by sampling candidate utterances from LLMs conditioned on the game context [12, 22]. Similarly, GoT utilizes LLMs to generate candidate questions at each decision point, making the game solvable while preserving the natural language nature of the actions.

Information-Seeking Ability of LLMs Although our primary focus is the Game of Twenty Questions, the problem is closely related to broader work on LLM information seeking. In underspecified tasks, an LLM may initially lack key information and must ask clarification questions to infer the user's intent [25, 51, 48] or elicit user preferences [17, 31]. In this sense, Twenty Questions provides a controlled abstraction of a central challenge in information seeking: selecting queries that efficiently identify the underlying state. Prior methods for improving information seeking include

prompting approaches such as SC and ToT [47, 50, 8], as well as heuristic query-selection methods based on expected information gain [16, 37], including UoT [20]. MCTS-based methods can also be adapted to such environments under a fixed distribution [9], but they do not directly address adversarial imperfect-information settings where such a distribution cannot be assumed. Benchmarks for evaluating information seeking have been proposed for under-specified tasks [30], medical diagnosis [32, 6], and troubleshooting [38]. These settings motivate our use of binary-question games: while less general than open-ended interaction, binary questions provide enough structure for principled solving and evaluation, and can already model these realistic scenarios.

Classical Search and Decision Trees. SLS builds on classical decision-tree optimization, where an unknown item is identified through sequential binary tests [28]. In particular, identifying a target item using the fewest questions is closely related to set cover and the minimum-test-set problem [11]. Adversarial variants of Twenty Questions have also been studied in the Renyi–Ulam framework [39, 43], which allows the answerer to provide a bounded number of false answers. In contrast, SLS assumes truthful answers, with adversarial behavior arising from the selection of the hidden item. Building on these classical foundations, our work focuses on applying game-theoretic optimization to natural language questions that cannot be exhaustively enumerated. We discuss these connections further in Appendix D.

3 Problem Formulation

We introduce an adversarial formulation of the Game of Twenty Questions referred to as *Strategic Language Search* (SLS). An SLS game is played over a finite set $\mathcal{S}$ of n distinct items between two competitive zero-sum players, the *Item Chooser* and the *Questioner*. The finiteness of $\mathcal{S}$ is deliberate: it allows us to enumerate all possible target items and evaluate worst-case performance exactly. We assume a countable set of binary, i.e., true-false, questions $\mathcal{Q}$, where each question is uniquely specified by a finite-length natural language string. We define $f : \mathcal{Q} \times \mathcal{S} \to \{0, 1\}$, where $f(q, s)$ denotes the *answer* to $q \in \mathcal{Q}$ for item $s \in \mathcal{S}$. A straightforward but degenerate choice of $\mathcal{Q}$ is the set of all possible binary questions, denoted by $\mathcal{Q}_\infty$. One textual representation of $\mathcal{Q}_\infty$ is the set of boolean-form questions such as "Is the item Item 1 or Item 3, but not Item 2?"

An SLS is defined by $(\mathcal{S}, \mathcal{Q}, f)$ and proceeds in two phases. In the first phase, the Item Chooser privately selects a single item $s^* \in \mathcal{S}$. In the second phase, the Questioner elicits information about s^* by sequentially asking and obtaining answers to a series of questions $q_1, a_1, \ldots, q_T, a_T$ where $q_t \in \mathcal{Q}$ and $a_t = f(q_t, s^*)$, i.e., the answer to q_t for s^*.

The history of questions and answers up to and including time $t \geq 0$ is given by $H_t = (Q_t, A_t)$, where $Q_t = (q_1, \ldots, q_t), A_t = (a_1, \ldots, a_t)$ such that $Q_t(\tau) = q_\tau$ and $A_t(\tau) = a_\tau$. Given history H, we denote by $H'(q, a)$ the new history when a new question q is asked with answer a. The Questioner selects its questions online, i.e., q_t may depend on observed history H_{t-1}. The length of history H is denoted by $|H|$. We denote the set of items consistent with some history $H = (Q, A)$ as

$$S(H) = \{s \in \mathcal{S} | \forall \tau \in [|H|], f(Q(\tau), s) = A(\tau)\} \subseteq \mathcal{S}.$$

Note that $S(H)$ is never empty, as it must contain at least s^*. The game ends when the Questioner can identify s^* with certainty, i.e., $|S(H)| = 1$, after which the Item Chooser (resp. Questioner) receives a reward (resp. cost) of $|H|$. The Questioner aims to find a strategy (or equivalently, policy) that minimizes its expected cost regardless of how the Item Chooser chooses s^*, optimizing its worst-case performance. This corresponds to the *NE* of a zero-sum game and will typically be randomized. We will define the solution concept formally after stating several assumptions and variants.

Assumption 3.1. Each question-item pair $(q, s) \in \mathcal{Q} \times \mathcal{S}$ has a unique answer $f(q, s)$ independent of history.

Assumption 3.2. After fixing s^*, the Item Chooser cannot lie with regard to its answers.

Assumption 3.3. $\mathcal{S}$, $\mathcal{Q}$, and f are common knowledge.

Assumption 3.4. Constant time oracle access to f.

Assumption 3.5. For every pair of distinct items $s, s' \in \mathcal{S}$, there exists some $q \in \mathcal{Q}$ where $f(q, s) \neq f(q, s')$.

Assumptions 3.1 and 3.2 are implicit assumptions baked into the definition of f while Assumption 3.5 is a technical assumption imposed on $\mathcal{Q}$ and f. It ensures that there exists a strategy that allows the

Questioner to find the correct item in finite time. Note that it is trivially satisfied when $\mathcal{Q}$ includes "identity" questions such as "is the item 's'?".

Example 1. *Consider the SLS with $\mathcal{S}$ and $\mathcal{Q}$:*

$$\left\{\begin{array}{l} s^{(1)} : \text{'Oppenheimer'} \\ s^{(2)} : \text{'Alan Turing'} \\ s^{(3)} : \text{'A Beautiful Mind'} \end{array}\right\}, \left\{\begin{array}{l} q^{(1)} : \text{'Related to codes?'} \\ q^{(2)} : \text{'Is it a movie?'} \\ q^{(3)} : \text{'Is it a person?'} \end{array}\right\}$$

Then $f(q^{(i)}, s^{(j)}) = 0$ if $j = i$, 1 otherwise.

It is easy to verify that Assumption 3.5 holds in Example 1. By selecting $q^{(1)}$ and $q^{(2)}$ in sequence, one can guarantee that the Questioner asks no more than 2 questions. The best the Item Chooser can do against this is to select $s^* = s^{(2)}$ or $s^{(3)}$, since doing so guarantees that there are always two items consistent after the first question, i.e., $|S(H_1)| = 2$. However, the Questioner can do better by choosing the first question uniformly at random. This guarantees that regardless of the choice of s^*, there is $1/3$ probability that the item can be determined with exactly one question, thus the expected number of questions asked is $1/3 \cdot 1 + 2/3 \cdot 2 < 2$. This strategy minimizes the worst-case cost regardless of what s^* the Item Chooser chose. In this example, the optimal strategy for the Questioner is symmetric due to the circular symmetric nature of $\mathcal{Q}$ and f. This is not true for more general $\mathcal{Q}, f$.

SLS has so far been presented mainly as a combinatorial problem. Indeed, when $\mathcal{Q}$ is finite, we have:

Theorem 3.6. *Given a known (deterministic) s^*, deciding if there exists a subset of k questions $Q \subseteq \mathcal{Q}$ such that $S(H) = \{s^*\}$ is NP-complete.*

The proof is a standard reduction from set cover and is included in Appendix C. In game-theoretic parlance, this implies that the *best response* of the Questioner to any deterministic Item Chooser strategy is hard to compute.

At the other extreme, we also observe that SLS recovers the classical binary-search solution when the Questioner can realize arbitrary binary partitions of the remaining candidate set. In particular, suppose $|\mathcal{S}| = 2^k$ and, at every round t, there exists a question q_{t+1} such that $|S(H'_{t-1}(q_t, 0))| = |S(H'_{t-1}(q_t, 1))|$. Then repeatedly selecting such an "even-split" question identifies the target in exactly k rounds. This is worst-case optimal since a binary question can reduce the candidate set by at most a factor of two, giving a lower bound of $\log_2 |\mathcal{S}| = k$ questions.

SLS-Restricted (SLSR) Vanilla SLS can be cumbersome to analyze for large n and $\mathcal{Q}$. We therefore introduce SLSR, a restricted variant in which questions are generated from the remaining item set $S(H)$. Formally, an SLSR is given by $(\mathcal{S}, \mathcal{Q}, f, g)$, where $g : 2^{\mathcal{S}} \backslash \emptyset \to 2^{\mathcal{Q}} \backslash \emptyset$ maps each nonempty $S \subseteq \mathcal{S}$ to a nonempty set of at most $m \geq 1$ questions. The rules and payoffs are identical to SLS, except that the Questioner must choose $q_t \in g(S(H_{t-1}))$. To ensure progress, we strengthen Assumption 3.5 by requiring g to return at least one question that strictly reduces the consistent item set. Since restricting the Questioner's actions cannot improve its optimal performance, SLSR provides a lower bound on performance in SLS, aligning with our focus on worst-case performance.

Assumption 3.7. For every subset $S \subseteq \mathcal{S}$ with at least two items, there exists some $q \in g(S)$ and a pair of distinct items $s, s' \in S$ where $f(q, s) \neq f(q, s')$

Weighted-SLS (WSLS) In many real-world settings, some items are more important and should be prioritized during information seeking. For example, delays in identifying life-threatening conditions can be especially costly. We therefore introduce weighted-SLS, defined by $(\mathcal{S}, \mathcal{Q}, f, w)$, where $w : \mathcal{S} \to \mathbb{R}^+$. Given selected item s^*, the Questioner's cost is redefined as $w(s^*) \cdot |H|$, penalizing prolonged interactions more heavily for high-weight items. Weighted SLSR (WSLSR), defined by $(\mathcal{S}, \mathcal{Q}, f, g, w)$, and the weighted non-adversarial game of Twenty Questions are defined analogously.

Example 2. *SLSR $(\mathcal{S}, \mathcal{Q}, f, g)$ with $\mathcal{S}, \mathcal{Q}, f$ identical to Example 1, $g(S) = \{q^{(1)}, q^{(2)}\}$ when $S = \mathcal{S}$ and $\mathcal{Q}$ otherwise.*

In Example 2, the first question is restricted to be either $q^{(1)}$ or $q^{(2)}$. Regardless of how the Questioner randomizes between them, the Item Chooser can always select $s^* = s^{(3)}$; this guarantees that $a_1 = 1$ and hence $|S(H_1)| = 2$. Thus, the Questioner requires at least 2 questions in total.

Example 3. *WSLS* $(\mathcal{S}, \mathcal{Q}, f, w)$ *with* $\mathcal{S}, \mathcal{Q}, f$ *identical to Example 1,* $w(s^{(1)}) = 3, w(s^{(2)}) = w(s^{(3)}) = 2$.

In Example 3, the Questioner can select $q^{(1)}, q^{(2)}, q^{(3)}$ with probability $3/4, 1/8, 1/8$ as q_1. This ensures the expected cost is no greater than $15/4$, whereas selecting q_1 uniformly incurs a cost of 5 when $s^* = s^{(1)}$ is chosen.

Remark 3.8. In SLS and its variants, the game only ends when the Questioner is perfectly sure of the s^*. An alternative formulation permits the Questioner to explicitly guess the item at any stage, with penalties for incorrect guesses. We chose the current formulation since it captures the constraints in high-stakes environments better, e.g., in medical diagnosis, one wishes to rule out all other possibilities before moving on to treatment.

Defining SLS via Large Language Models. For many applications, $\mathcal{S}$ is too large for us to realistically hand-curate a set of $\mathcal{Q}$, noting that we would like to avoid "unnatural" questions that involve long, complicated boolean expressions. Therefore, in this paper we define $\mathcal{Q}$, f, and g implicitly using Large Language Models (LLMs). Specifically, $\mathcal{Q}$ is the set of questions that an LLM can propose asking based on certain prompts, and f is the response of an LLM to a prompt of q when asked about s^*. For instance, `Is Oppenheimer a movie?`. We wish to emphasize that f can be implemented by any oracle, such as human experts. We chose to implement it via LLMs for the ease of experimentation. Furthermore, g in SLSR and WSLSR can be defined as the output of an LLM with respect to a prompt asking for $m \geq 1$ questions after defining the rules of SLS and $S(H)$, the items remaining. An example of this is `Currently the set of possible items is {Oppenheimer, Alan Turing}. Propose m best questions.`

Assumption 3.9. The LLM for f makes no mistakes when used as an oracle for $f(q, s^*)$.

Assumptions 3.9 and 3.1 enable the practical usage of LLMs as a black box for defining SLS. For most LLMs, Assumption 3.4 is also satisfied, though the constant factor may be large. We further discuss them in Appendix H.

When LLMs are used to construct or play the game, token usage introduces an additional notion of cost. In this work, however, we focus on interaction cost, as this is often dominant in the settings we consider. For example, in medical diagnosis, a single turn may correspond to an additional diagnostic test, requiring clinician time, equipment use, laboratory materials, and imposing additional burden on the patient. In such cases, reducing the number of real-world interactions is more consequential than reducing inference-time token usage. We therefore define player utilities primarily in terms of interaction turns, while reporting token statistics separately.

4 SLS as Zero-sum Extensive Form Games

A vanilla SLS $(\mathcal{S}, \mathcal{Q}, f)$ can be represented as a two-player zero-sum extensive-form game (EFG) with imperfect information. An EFG is a rooted game tree whose vertices correspond to the game's entire history and whose edges represent available actions to the player-to-move at that vertex. EFGs are endowed with information sets (infosets) that partition vertices of the same player; vertices in the same infoset are indistinguishable to that player, capturing the notion of imperfect information.

In the EFG formulation, the Item Chooser only takes an action at the root, after which the Questioner asks up to $n-1$ questions in sequence.[1] Every state apart from the root can be uniquely identified by the tuple (s^*, H) which describes item s^* chosen and a (potentially empty) history of past questions and answers. Leaf vertices are those where either (i) $|H| = n-1$, where the maximum number of questions are asked, or (ii) $|S(H)| = 1$, where s^* is identified. At a leaf, the reward (resp. cost) to the Item Chooser (resp. Questioner) is $|H|$. All non-leaf vertices with the same history H belong to the same infoset, which we denote by $I(H)$. Figure 1 illustrates the SLS for Example 1.

A deterministic strategy π of the questioner is given by a mapping from every infoset $I(H)$ to some question $q \in \mathcal{Q}$. Following π when s^* is chosen yields the following history $H_0, \ldots H_T$, given by

$$q_{t+1} = \pi(I(H_t)), a_{t+1} = f(\pi(I(H_t)), s^*),$$

[1]This keeps the game tree finite, and is justified since asking the same question more than once is suboptimal.

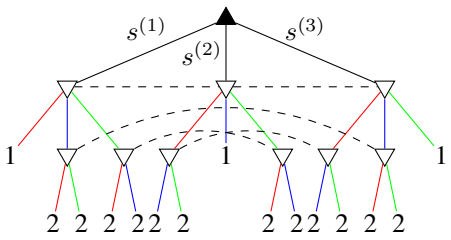

Figure 1: EFG representation of Example 1. The root node ▲ belongs to the Item Chooser and the other nodes ▽ belong to the Questioner. Edges in black correspond to the choice of item s^*. Colored edges in red, blue, and green refer to questions $q^{(1)}$, $q^{(2)}$, and $q^{(3)}$ respectively. Vertices connected by dotted lines belong to the same infoset. Payoffs to the Item Chooser are shown in the leaves. We omit edges where the same question is asked more than once; these are never optimal.

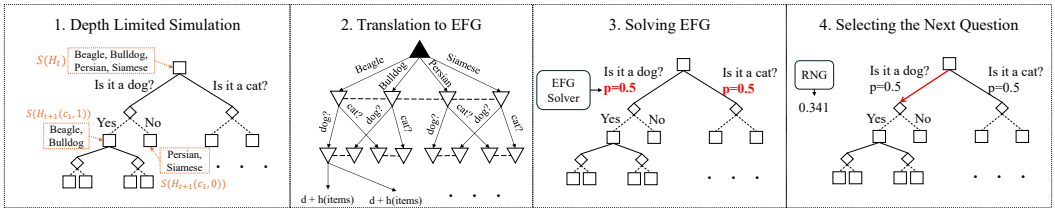

Figure 2: **An overview of GoT.** To choose the question at time step t, we (1) perform depth-limited simulation (2) construct a subgame (3) solve it for a local strategy which (4) informs the choice of the question. Steps 1 to 3 devise the strategy, while step 4 plays the game.

where T is the time the game ends, i.e., $|S(H_T)| = 1$. The corresponding utility for the Item Chooser (resp. cost for the Questioner) at the end of the game is $u(\pi, s^*) = |H_T|$.

Let the set of all deterministic strategies be Π, Δ_Π the probability simplex over Π, $\Delta_\mathcal{S}$ the probability simplex over $\mathcal{S}$. Then the expected utility under random policies $x \in \Delta_\Pi$ and $y \in \Delta_\mathcal{S}$ for Questioner and Item Chooser respectively is (with some abuse of notation) denoted by

$$u(x, y) = \mathbb{E}_{\pi \sim x, s^* \sim y} \left[u(\pi, s^*) \right].$$

The NE of this zero-sum game is given by the solution to the bilinear saddle-point problem

$$\min_{x \in \Delta_\Pi} \max_{y \in \Delta_\mathcal{S}} u(x, y) = \max_{y \in \Delta_\mathcal{S}} \min_{x \in \Delta_\Pi} u(x, y) \tag{1}$$

where equality holds as a consequence of Von Neumann's minimax theorem[2] [44]. The solution (x^*, y^*) yields optimal randomized strategies for each player. In particular, x^* is the *distribution* over Questioner policies that is robust against the worst-case choice of s^*. EFGs for SLSR, WSLS and WSLSR may also be constructed by one or both of (i) restricting the actions available at each infoset to $g(S(H))$ (ii) adjusting leaf payoffs to depend on w and s^*.

In general, two-player zero-sum EFGs with perfect recall (including SLS and its variants) can be solved in time polynomial to the size of the game tree. Some classical methods include linear programming [45] and counterfactual regret (CFR) minimization [53]. Note that the definition of strategy given for the Questioner was in normal (also known as strategic or matrix) form. This leads to an exponentially sized action space, so in practice solvers usually operate (implicitly) in the strategically equivalent behavioral or *sequence form* [45]. Efficient game-solving and strategy representation in EFGs is a well-studied topic and beyond the scope of this paper, though we give a brief overview in Appendix B. Instead, we will simply employ off-the-shelf solvers [33, 27].

5 Game of Thought and Subgame Search

While ideally we wish to solve for the NE of an SLS game, with the space of natural language questions being too large to exhaustively explore, for practicality we instead target the NE of SLSR

[2]This holds for finite $\mathcal{Q}$ and SLSR.

with a small candidate question set size m. This restriction makes game-theoretic optimization tractable, but limits the resulting strategies to the questions proposed by the candidate generator. Without additional assumptions on question coverage, optimality within SLSR does not imply optimality in SLS. Nonetheless, even under this restriction, explicitly constructing and solving the *full* (W)SLSR game tree is impractical for larger hypothesis spaces, as it entails querying an LLM at every infoset to propose questions and provide answers, with each call incurring multi-second latency. To address this challenge, GoT employs an iterative, on-demand strategy construction paradigm, computing strategies only when an infoset is reached. Inspired by subgame search techniques used in poker [2, 3], GoT constructs a subgame rooted at the current infoset and truncates it to a fixed depth d. The leaf nodes of this subgame are evaluated using heuristic functions. This truncated subgame is then solved to obtain a local strategy for the questioner, which informs the selection of the next question. An overview is provided in Figure 2.

1. Depth-Limited Simulation We begin by exploring the possible future outcomes and questions for the Questioner. At infoset $I(H_t)$, a set of candidate questions $g(S(H_t))$ is generated using an LLM. For each candidate, we use an LLM as f to identify the set of items for which the answer to c_i is yes, denoted by $Y(S(H_t), c_i)$ where $Y(S, q) = \{s \in S | f(q, s) = 1\}$ for some $S \subseteq \mathcal{S}$, and the complement set denoted by $\bar{Y}(S(H_t), c_i) = S(H_t) \backslash Y(S(H_t), c_i)$. This pair of sets represents the two possible outcomes of asking the candidate question c_i at step t, with $Y(S(H_t), c_i)$ (resp. $\bar{Y}(S(H_t), c_i)$) representing the set of items consistent with the history $H'_t(c_i, 1)$ (resp. $H'_t(c_i, 0)$). The corresponding infosets $I(H'_t(c_i, 1))$, $I(H'_t(c_i, 0))$ are constructed based on this pair of outcomes. This process is repeated to construct the infosets for d steps, resulting in a simulation tree as seen in step 1 of Figure 2. Note that this step is similar to the question simulation and generation step of the UoT framework, and incurs a similar number of LLM calls when the same d and m are used.

2. Translation to EFG We then construct a subgame based on the simulation tree. The tree is converted into an EFG representation of the truncated subgame, as shown in step 2 of Figure 2. Note that the subgame begins with the Item Chooser "re-choosing" a new distribution over $S(H_t)$ at each iteration even if $I(H_t)$ is not at the beginning of the game (i.e., $t \neq 0$). Each leaf node l in the game tree is assigned a payoff of $d(l) - 1 + h(l)$ for the Item Chooser (and the negative of that for the Questioner) where $d(l)$ is the depth of node l in the tree, corresponding to the number of questions asked since the start before reaching l. h is a heuristic function that estimates the number of questions that remain to be asked for the Questioner to identify the correct item. In our experiments for SLSR we set $h(l) := \log_2(|S(l)|)$ as an optimistic estimate of the remaining turns.

3. Extensive Form Game Solving Use any EFG solver to solve the subgame. In this paper, we used LiteEFG's [33] implementation of CFR [53] to find an approximate NE.

4. Selecting the Next Question To choose the question to ask at step $t + 1$, one question q_{t+1} is sampled from $g(S(H_t))$ using the Questioner strategy from the NE. We move to the infoset $I(H'_t(q_{t+1}, 0))$ if $f(q_{t+1}, s^*) = 0$, and $I(H'_t(q_{t+1}, 1))$ otherwise.

These four steps are repeated until the process reaches an infoset $I(H_n)$ with $S(H_n) = s^*$. The computational cost of GoT is dominated by the simulation step, primarily due to LLM latency, while the other steps incur negligible overhead. Because UoT and GoT share a similar simulation process, their overall computational costs are comparable. Thus, GoT can be viewed as a robust extension of UoT that introduces little additional computational overhead.

Step 2 of GoT augments the subgame to allow the Item Chooser to "re-choose" a new distribution over s^*. Doing this rather than sticking to the distribution chosen by the Item Chooser at the start of the game seemingly accords more power to the Item Chooser. This is in fact a game solving technique known as *safe* subgame search [35, 23], which preserves a performance guarantee when we resolve for the questioner's strategy at each iteration. In contrast, an algorithm that is not safe may design a strategy that improves performance inside the subgame at each iteration but worsens the global performance. We further discuss this in Appendices E and G.1.

6 Experiments and Results

We are interested in answering the following question: *Does GoT improve the **worst-case** performance compared to other methods in all settings?* We also present additional results in Appendix G.

6.1 Experimental Setup

Datasets and Settings Each dataset is a collection of distinct items. We constructed several datasets based on prior studies: (i) *Common*: A set of 136 items built upon the dataset collected for UoT [20] (ii) *Breeds*: A set of 25 popular cat and dog breeds. (iii) *S128*: A set of 128 items, consisting of 2 weapons, 6 scientists, 24 dishes, and 96 animals. In addition to 20Q, we also present results in two practical domains. **Medical Diagnosis (MD)**: A doctor seeks to diagnose a patient by asking about the symptoms they are experiencing. We use *DXBench (DX)* [6], and randomly select 100 out of the 461 unique diseases in the dataset as the hypothesis space. **Troubleshooting (TS)** A car mechanic attempts to help a customer diagnose the car fault they are experiencing by asking about the symptoms. We obtained 59 unique car faults from *FloDial (FD)* [38] to serve as the hypothesis space. Details are given in Appendix I.

Baselines We compare methods from three broad categories. First, we consider two *pure prompting-based methods*: Direct Prompting (DP), where the LLM directly generates the next question, and Direct Choice (DC), where the LLM selects from a set of candidate questions. Second, we consider two *non-game-theoretic search-based methods*, UoT [20] and Monte Carlo Tree Search for Information Seeking Questions (MISQ) [7], a recent MCTS-based approach. These are compared with our adversarial search method, GoT. Details are provided in Appendix F. Unless otherwise specified, we set $d = m = 3$ for both UoT and GoT.

Metrics We primarily report worst-case performance, measured by $\max_{s \in \mathcal{S}} \mathbb{E}(|H^s|)$, where H^s is the interaction history when s is the chosen item. Token statistics are reported in Appendix G.8.

Models and Prompts We mainly use GPT-4.1 [36] (`gpt-4.1-2025-04-14`) and Qwen2.5-72B-Instruct [49]. For 20Q, we considered two question-sampling prompts. The *even* prompt, adopted from [20], instructs the LLM to ask questions that split items as evenly as possible. However, it sometimes produced unnatural questions, e.g., "Does this item begin with a letter from A–M?", which are unsuitable for realistic settings. We therefore use *natural* prompts, which discourage such questions, for all results below. This issue did not arise in MD or TS, possibly because the LLMs are explicitly instructed to act as a doctor or mechanic. Prompts are shown in Appendix K, with examples of sampled questions in Appendix J.

Accounting for Randomness There are two sources of randomness in our experiments. **(1)** When an LLM is used to generate questions, the set of candidate questions $\mathcal{Q}(S(H_t))$ may differ across runs. To provide a fair comparison between methods, we first play the game using GoT and cache the questions in the simulation tree. The cached questions are reused for DC, UoT, and MISQ. **(2)** Given a fixed set of generated candidate questions, GoT yields a nondeterministic strategy. We average over ten plays using the same strategy and report the average performance. DC, DP, and MISQ are similarly random and thus also repeated ten times. UoT is deterministic, and thus is not repeated.

6.2 Worst-Case Performance

Unweighted Variant We evaluate all methods under the unweighted variant, where performance is measured by worst-case interaction length. As shown in Table 1, GoT consistently improves over search-based baselines such as UoT and MISQ. The substantially weaker performance of pure prompting-based methods such as DP and DC further shows that *Twenty Questions, despite its simple rules, requires nontrivial strategic reasoning*. This highlights the effectiveness of supporting LLM reasoning with external, game-theoretic optimization as GoT does. We also conducted paired t-tests on ten runs on all five datasets using GPT 4.1 in the SLSR setting to compare the performance of GoT and UoT methods. In each run, a different set of questions is used. The results are shown in Table 2. The paired t-test results indicate that GoT significantly outperforms UoT ($p < 0.05$).

Method	Unweighted Variant					Weighted Variant			
	Common	S128	Breeds	DX	FD	Common	Breeds	DX	FD
GPT 4.1									
GoT	<u>10.2</u>	<u>11.8</u>	<u>7.4</u>	<u>12.2</u>	<u>7.9</u>	<u>152.1</u>	<u>23.2</u>	<u>78.3</u>	<u>61.4</u>
UoT	11.0	13.0	9.0	13.0	9.0	227.4	32.1	110.0	81.0
MISQ	13.9	14.5	8.6	14.3	10.3	193.3	56.4	125.1	91.3
DP	13.8	16.2	7.8	16.8	12.7	224.0	36.9	116.0	90.1
DC	12.9	14.6	9.3	16.2	11.6	199.2	41.1	126.4	97.4
Qwen 2.5 72B Instruct									
GoT	<u>10.0</u>	<u>10.8</u>	<u>6.6</u>	<u>10.5</u>	<u>7.5</u>	<u>151.9</u>	<u>32.3</u>	<u>73.6</u>	<u>62.3</u>
UoT	11.0	12.0	8.0	12.0	9.0	228.5	47.0	99.0	74.0
MISQ	12.5	18.6	9.1	14.2	10.1	197.8	49.1	121.5	87.9
DP	12.7	19.2	8.0	12.4	10.7	235.7	47.8	114.3	80.0
DC	12.7	17.9	7.8	13.3	10.5	225.1	45.7	86.1	113.6

Table 1: **Worst-case interaction length for each method in various settings**. Lower is better. Performance shown for the unweighted variant is $\max_{s\in\mathcal{S}} |H^s|$. Performance shown for the weighted variant is $\max_{s\in\mathcal{S}}(w(s)\cdot|H^s|)$. Best performance for each setting is <u>underlined</u>.

Dataset	GoT	UoT	t-Statistic	p-Value
Common	10.2	11.3	-5.14	0.00030
S128	12.0	13.1	-7.88	0.00001
Breeds	6.8	8.0	-4.68	0.00058
DX	12.0	12.9	-8.27	0.00001
FD	7.86	8.7	-5.02	0.00036

Table 2: **Paired t-test statistics comparing UoT and GoT.** Samples are obtained across 10 runs for each dataset. For both UoT and GoT, $d = m = 3$. The model used is GPT 4.1.

Weighted Variant We also evaluate all methods under the weighted setting. GoT formulates the problem as a WSLSR game and uses the subgame payoff $\max_{s\in S(l)} w(s)\cdot(d(l)+\log_2|S(l)|)$ to approximate the objective $w(s^*)\cdot|H|$. For all methods, we modify the question-sampling prompt to include item weights and the corresponding payoff definition. As shown in Table 1, *GoT outperforms UoT across all settings, with larger gains than in the unweighted variant*. Notably, UoT's advantage over prompting-based methods such as DP and DC becomes less pronounced. This is likely because UoT maximizes information gain without accounting for item weights, and may therefore prefer questions that split the item set evenly over questions that isolate high-weight items.

6.3 Average Performance of Unweighted Variant

While we primarily focus on the worst-case performance, it is important to also examine the trade-offs with the average performance in actual applications. We present this comparison below.

Average Performance on Dataset Distribution We examine the average performance on DX under sampled multinomial priors P^o. The average conversation length is calculated as $L_{\text{avg}} = \sum_{s\in\mathcal{S}} P^o(s)\cdot|H^s|$. This measures the expected conversation length when diseases are drawn according to P^o. We sample 50 priors from $Dir(kC)$, where $C\in\mathbb{Z}^{100}_{\geq 0}$ denotes disease occurrence counts in the dataset. In Figure 3, we plot L_{avg} alongside L_{worst} to show the trade-off between average and worst-case performance. We also include a best-response (BR) strategy, where the Questioner optimizes against a publicly known prior P^o, serving as a theoretical lower bound. Details are provided in Appendix F.4. The results show two main trends. First, GoT and UoT achieve similar average-case performance, with only a small gap relative to their improvement over DP/DC and BR's improvement over both. Second, GoT achieves a substantial worst-case improvement over UoT, comparable in magnitude to UoT's average-case gain over DP. Overall, *GoT provides meaningful worst-case gains with little loss in average-case performance relative to UoT.*

Average Performance on Unfavorable Priors We further compare UoT and GoT under priors designed to be unfavorable to each method. For each method, we sample priors from $Dir(k\alpha)$, where

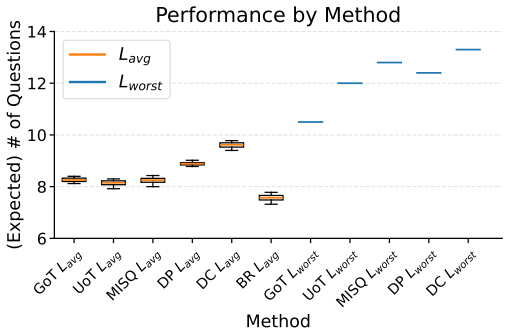

Figure 3: **Average and worst-case performance on DX.** The model used is Qwen 2.5 72B Instruct, with k set to 1. Variance in the average performance is due to different samples of P^o, not randomness in the methods.

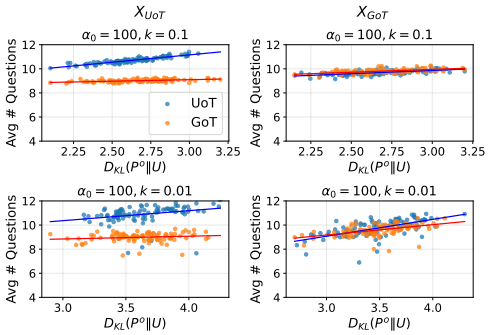

Figure 4: **Average performance in various family of distributions** Left column corresponds to priors unfavorable to UoT, while right column corresponds to priors unfavorable to GoT. X-axis is the KL divergence from uniform distribution.

$\alpha_i = 1$ for all $i \neq 0$, and α_0 is increased to concentrate probability mass on a single item. The index 0 is chosen to correspond to the item that induces the worst-case conversation length for that method. We draw 100 priors for each configuration and compare L_{avg}. As shown in Figure 4, when priors are concentrated around UoT's worst-case item, UoT degrades substantially more than GoT. In contrast, when priors are concentrated around GoT's worst-case item, the average-case gap is much smaller. These results suggest that *GoT is more robust than UoT to unfavorable item distributions.*

7 Limitations and Conclusion

We formalized the definition of SLS and its variants, and proposed GoT, a simple yet effective approach to play SLSR. We show that GoT is able to improve the worst-case performance as compared to prior methods. Several directions remain open. GoT optimizes over a fixed set of candidate questions, but how its strategic feedback can be used to improve LLM question generation remains to be explored. Furthermore, extending GoT to open-ended settings while retaining rigorous evaluation is an important direction for future work. We further discuss these in Appendix A.

8 Acknowledgements

This research is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG3-PhD-2024-08-057), as well as the National University of Singapore under the Start-Up Grant Scheme.

References

[1] Justin Bishop, Jaylen Burgess, Cooper Ramos, Jade B. Driggs, Tom Williams, Chad C. Tossell, Elizabeth Phillips, Tyler H. Shaw, and Ewart J. de Visser. CHAOPT: A testbed for evaluating human-autonomy team collaboration using the video game Overcooked!2. In *2020 Systems and Information Engineering Design Symposium (SIEDS)*, pages 1–6, 2020.

[2] Noam Brown and Tuomas Sandholm. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424, 2018.

[3] Noam Brown and Tuomas Sandholm. Superhuman AI for multiplayer poker. *Science*, 365(6456):885–890, 2019.

[4] Neil Burch, Michael Johanson, and Michael Bowling. Solving imperfect information games using decomposition. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, page 602–608, 2014.

[5] Valerio Capraro, Roberto Di Paolo, Matjaž Perc, and Veronica Pizziol. Language-based game theory in the age of artificial intelligence. *Journal of The Royal Society Interface*, 21(212), 2024.

[6] Junying Chen, Chi Gui, Anningzhe Gao, Ke Ji, Xidong Wang, Xiang Wan, and Benyou Wang. CoD, towards an interpretable medical agent using chain of diagnosis. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 14345–14368, 2025.

[7] Harshita Chopra and Chirag Shah. Feedback-aware monte carlo tree search for efficient information seeking in goal-oriented conversations. *arXiv preprint arXiv:2501.15056*, 2025.

[8] Deepro Choudhury, Sinead Williamson, Adam Goliński, Ning Miao, Freddie Bickford Smith, Michael Kirchhof, Yizhe Zhang, and Tom Rainforth. BED-LLM: Intelligent information gathering with LLMs and Bayesian experimental design. *arXiv preprint arXiv:2508.21184*, 2025.

[9] Ziwei Deng, Mian Deng, Chenjing Liang, Zeming Gao, Chennan Ma, Chenxing Lin, Haipeng Zhang, Songzhu Mei, Siqi Shen, and Cheng Wang. PlanU: Large language model reasoning through planning under uncertainty. *arXiv preprint arXiv:2510.18442*, 2025.

[10] Joseph Farrell and Matthew Rabin. Cheap talk. *Journal of Economic Perspectives*, 10(3):103–118, 1996.

[11] Michael R Garey and David S Johnson. *Computers and intractability*, volume 29. wh freeman New York, 2002.

[12] Ian Gemp, Yoram Bachrach, Marc Lanctot, Roma Patel, Vibhavari Dasagi, Luke Marris, Georgios Piliouras, Siqi Liu, and Karl Tuyls. States as strings as strategies: Steering language models with game-theoretic solvers. *arXiv preprint arXiv:2402.01704*, 2024.

[13] Itzhak Gilboa and David Schmeidler. Maxmin expected utility with non-unique prior. *Journal of mathematical economics*, 1989.

[14] Herbert Gintis, Eric Alden Smith, and Samuel Bowles. Costly signaling and cooperation. *Journal of Theoretical Biology*, 213(1):103–119, 2001.

[15] Sally A Goldman and Michael J Kearns. On the complexity of teaching. *Journal of Computer and System Sciences*, 50(1):20–31, 1995.

[16] Gabriel Grand, Valerio Pepe, Jacob Andreas, and Joshua B. Tenenbaum. Loose lips sink ships: Asking questions in battleship with language-informed program sampling. *arXiv preprint arXiv:2402.19471*, 2024.

[17] Kunal Handa, Yarin Gal, Ellie Pavlick, Noah Goodman, Jacob Andreas, Alex Tamkin, and Belinda Z Li. Bayesian preference elicitation with language models. *arXiv preprint arXiv:2403.05534*, 2024.

[18] Lars Peter Hansen and Thomas J Sargent. Robust control and model uncertainty. *American Economic Review*, 2001.

[19] Daniel Houser and Erte Xiao. Classification of natural language messages using a coordination game. *Experimental Economics*, 14:1–14, 2011.

[20] Zhiyuan Hu, Chumin Liu, Xidong Feng, Yilun Zhao, See-Kiong Ng, Anh Tuan Luu, Junxian He, Pang Wei Koh, and Bryan Hooi. Uncertainty of thoughts: Uncertainty-aware planning enhances information seeking in LLMs. In *Proceedings of the 38th Conference on Neural Information Processing Systems*, NeurIPS '24, 2024.

[21] Bimal P Jain. Why is diagnosis not probabilistic in clinical-pathological conference (cpcs): Point. *Diagnosis*, 3(3):95–97, 2016.

[22] Benjamin Kempinski, Ian Gemp, Kate Larson, Marc Lanctot, Yoram Bachrach, and Tal Kachman. Game of thoughts: Iterative reasoning in game-theoretic domains with large language models. In *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems*, pages 1088–1097, 2025.

[23] Vojtech Kovarık, Dominik Seitz, and Viliam Lisy. Value functions for depth-limited solving in imperfect-information games. In *AAAI Reinforcement Learning in Games Workshop*, volume 132, pages 133–138, 2021.

[24] János Kramár, Tom Eccles, Ian M. Gemp, Andrea Tacchetti, Kevin R. McKee, Mateusz Malinowski, Thore Graepel, and Yoram Bachrach. Negotiation and honesty in artificial intelligence methods for the board game of Diplomacy. *Nature Communications*, 13, 2022.

[25] Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. CLAM: Selective clarification for ambiguous questions with generative language models. *arXiv preprint arXiv:2212.07769*, 2022.

[26] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626, 2023.

[27] Marc Lanctot, Edward Lockhart, Jean-Baptiste Lespiau, Vinicius Zambaldi, Satyaki Upadhyay, Julien Pérolat, Sriram Srinivasan, Finbarr Timbers, Karl Tuyls, Shayegan Omidshafiei, et al. OpenSpiel: A framework for reinforcement learning in games. *arXiv preprint arXiv:1908.09453*, 2019.

[28] Hyafil Laurent and Ronald L Rivest. Constructing optimal binary decision trees is np-complete. *Information processing letters*, 1976.

[29] David Kellogg Lewis. *Convention: A Philosophical Study*. Wiley-Blackwell, Cambridge, MA, USA, 1969.

[30] Belinda Z Li, Been Kim, and Zi Wang. QuestBench: Can LLMs ask the right question to acquire information in reasoning tasks? *arXiv preprint arXiv:2503.22674*, 2025.

[31] Belinda Z Li, Alex Tamkin, Noah Goodman, and Jacob Andreas. Eliciting human preferences with language models. *arXiv preprint arXiv:2310.11589*, 2023.

[32] Shuyue Stella Li, Vidhisha Balachandran, Shangbin Feng, Jonathan S. Ilgen, Emma Pierson, Pang Wei Koh, and Yulia Tsvetkov. MediQ: Question-asking LLMs and a benchmark for reliable interactive clinical reasoning. In *Advances in Neural Information Processing Systems*, volume 37, pages 28858–28888, 2024.

[33] Mingyang Liu, Gabriele Farina, and Asuman Ozdaglar. LiteEFG: An efficient python library for solving extensive-form games. *arXiv preprint arXiv:2407.20351*, 2024.

[34] Xiangyu Liu, Di Wang, Zhe Feng, and Aranyak Mehta. Scaling inference-time computation via opponent simulation: Enabling online strategic adaptation in repeated negotiation. *arXiv preprint arXiv:2602.19309*, 2026.

[35] Matej Moravcik, Martin Schmid, Karel Ha, Milan Hladik, and Stephen Gaukrodger. Refining subgames in large imperfect information games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[36] OpenAI. GPT-4.1, 2024.

[37] Wasu Top Piriyakulkij, Volodymyr Kuleshov, and Kevin Ellis. Active preference inference using language models and probabilistic reasoning. *arXiv preprint arXiv:2312.12009*, 2023.

[38] 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*, pages 4348–4366, 2021.

[39] Alfréd Rényi. On a problem of information theory. *A Magyar Tudományos Akadémia Matematikai Kutató Intézetének Közleményei*, 1961.

[40] Hong Ri, Xiaohan Kang, Mohd Nor Akmal Khalid, and Hiroyuki Iida. The dynamics of minority versus majority behaviors: a case study of the mafia game. *Information*, 13(3):134, 2022.

[41] Robert S. Siegler. The twenty questions game as a form of problem solving. *Child Development*, 48(2):395–403, 1977.

[42] Matthijs TJ Spaan. Partially observable markov decision processes. In *Reinforcement learning: State-of-the-art*, pages 387–414. Springer, 2012.

[43] Stanislaw M Ulam. Adventures of a mathematician. In *Mathematics*. Chapman and Hall/CRC, 2019.

[44] J v. Neumann. Zur theorie der gesellschaftsspiele. *Mathematische annalen*, 100(1):295–320, 1928.

[45] Bernhard Von Stengel. Efficient computation of behavior strategies. *Games and Economic Behavior*, 14(2):220–246, 1996.

[46] Abraham Wald. Statistical decision functions. In *Breakthroughs in Statistics: Foundations and Basic Theory*, pages 342–357. Springer, 1950.

[47] 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.

[48] Jingjing Xu, Yuechen Wang, Duyu Tang, Nan Duan, Pengcheng Yang, Qi Zeng, Ming Zhou, and Xu Sun. Asking clarification questions in knowledge-based question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1618–1629, 2019.

[49] Qwen An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxin Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yi-Chao Zhang, Yunyang Wan, Yuqi Liu, Zeyu Cui, Zhenru Zhang, Zihan Qiu, Shanghaoran Quan, and Zekun Wang. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[50] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of Thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 11809–11822, 2023.

[51] Xuan Zhang, Yang Deng, Zifeng Ren, See-Kiong Ng, and Tat-Seng Chua. Ask-before-plan: Proactive language agents for real-world planning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10836–10863, 2024.

[52] Yizhe Zhang, Jiarui Lu, and Navdeep Jaitly. Probing the multi-turn planning capabilities of LLMs via 20 question games. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1495–1516, 2024.

[53] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In *Advances in Neural Information Processing Systems*, volume 20, 2007.

A Discussion on Limitations and Future Work

Improving Question Quality GoT primarily focuses on strategy optimization given a fixed set of LLM-generated candidate questions at each decision point. However, the resulting strategies and gameplay outcomes could also be used to provide feedback to the LLM, enabling it to generate more effective questions in subsequent iterations. For instance, if certain items consistently require longer conversations to identify, this information could be used to prompt the LLM to generate questions that better distinguish these items from the remaining candidates. Developing principled methods for extracting such signals from the Questioner's strategy, and for communicating them back to the LLM during question generation, is an interesting direction for future work.

Extension to Open-Ended settings GoT restricts candidate questions to those with strictly binary answers, which may limit the applicability of our framework to more general domains. This restriction is a deliberate modeling choice: binary responses induce structured transitions, making both principled evaluation and solver-based optimization tractable. Extending the framework to open-ended responses is an important direction for future work, but it introduces substantial challenges. In particular, under our assumption of a truthful answerer who knows the target item, unrestricted open-ended questions would make the problem trivial, as the Questioner could simply ask "What is the item?". A meaningful extension would therefore require additional constraints on the information that can be revealed, as well as a principled treatment of partial or ambiguous answers. Although prior methods such as UoT consider open-ended questions, they reduce it to binary classifications and do not directly resolve these formulation challenges. We leave a more general treatment of open-ended information seeking for future work.

Computational Cost As discussed in the main paper, our intended applications for GoT involve settings where the cost of an additional real-world interaction may be substantially more expensive than the extra inference-time computation. This design philosophy of trading additional inference-time computation for stronger task performance is also shared by many inference-time algorithms [34, 50, 47, 20]. For real-time applications where latency is a primary concern, GoT's computational cost can be controlled by adjusting hyperparameters such as search depth and branching factor, albeit at the cost of potentially weaker worst-case performance. The computational burden can also be mitigated through engineering strategies such as caching and reuse, precomputing candidate questions, and solving subgames offline, thereby trading higher upfront computation for lower deployment-time latency. Designing and experimenting with these engineering techniques is left as future work.

B Strategy Representation

There are several ways to represent strategies (up to payoff equivalence) in imperfect information games with perfect recall. For computational reasons, the *sequence form* is often used. We briefly describe the other representations.

1. Normal/Strategic Form. This was what was presented in the paper. The set of deterministic strategies is the Cartesian product of actions at each infoset $\Pi = \prod_{I \in \mathcal{I}} \mathcal{A}(I)$. Thus, every policy $\pi \in \Pi$ tells us exactly what action to take at every information set that could be encountered. Randomized strategies are distributions over deterministic policies, i.e., Δ_Π. The size of Π is exponential in the number of infosets.
2. Reduced Normal Form. The reduced normal form prunes the number of normal form strategies by grouping together strategically equivalent ones. See any textbook in game theory, or the work by [45] for a more complete discussion. Let I be an infoset and $\alpha, \alpha' \in \mathcal{A}(I)$ be distinct actions in I. Let I' be an infoset that α' precedes. Then, normal form strategies that choose α in I have payoffs (regardless of opponent strategy) that are independent of what action is taken in I', since to enter I' at all would require choosing α'. Thus, two strategies π, π' that choose α in I but differ in actions taken in I' are strategically equivalent. The minimal set of strategies that are obtained by forming such equivalence classes forms the set of reduced-normal form strategies, which can be significantly smaller than normal form ones. As before, strategies can be randomized in reduced normal form as well. Unfortunately, the size of reduced normal form strategies can still be exponential in the number of infosets.

3. Behavioral Form. The behavioral strategies involve placing a *distribution* of actions (possibly not deterministic) at each information set, thus the size is linear in the total number of actions summed over all infosets, which in turn is no larger than the size of the game tree. It can be shown using *Kuhn's theorem* that under perfect recall, the set of behavioral strategies is strategically equivalent to (reduced) normal form strategies. Unfortunately, while behavioral strategies are intuitive and compact, optimizing in behavioral form is difficult as payoffs are non-linear in the strategy representation.
4. Sequence Form. The sequence form strategy alleviates the problem of assigning probabilities to sequences σ (essentially actions when the game has perfect recall) of taking a particular sequence in isolation from chance and other players. At each infoset I, the sequence form is essentially identical to the behavioral form (a probability simplex) except that they are normalized by the probabilities given by the infoset's parent sequence $\sigma(I)$, which is the last action (sequence) taken before reaching I — this is guaranteed to be unique because of perfect recall. Thus, the sequence form is the same size as behavioral strategies (except for an extra "empty sequence" ϕ set to 1.0 that is the parent of all initial infosets). The sequence form strategy space is sometimes known as the treeplex

$$\mathcal{X} = \left\{ x \in \mathbb{R}_+^N \middle| x[\phi] = 1; \sum_{\sigma = Ia} x[\sigma] = x[\sigma(I)] \; \forall \; I \in \mathcal{I} \right\}$$

where $N = 1 + \sum_{I \in \mathcal{I}} |I|$ and $\mathcal{I}$ is the set of infosets, Ia refers to the sequences ending with action i starting at infoset I, and ϕ is the empty sequence. The vertices of $\mathcal{X}$ correspond to (reduced) normal form strategies. Using the sequence form, we can write the Nash equilibrium of a zero-sum game as the bilinear saddle point problem

$$\min_{x \in \mathcal{X}} \max_{y \in \mathcal{Y}} x^T M y$$

where M is the sequence form payoff matrix. Observe that the objective is linear in both x and y. This saddle point can be found efficiently using a variety of methods, including first order methods such as the counterfactual regret minimization [53], linear programming or other first order methods (e.g., mirror prox). The library which we use [33] uses counterfactual regret minimization.

C Proof of Theorem 3.6

This problem is equivalent to the following set-intersection problem.

Set intersection problem. Let $\mathcal{S}$ be a finite set and $\mathcal{K} \subseteq 2^{\mathcal{S}}$ a set of sets. Given some fixed $s^* \in \mathcal{S}$, we want to find if there is a set $\mathcal{J} \subseteq \mathcal{K}$ such that $|\mathcal{J}| \leq k$ and $\bigcap_{J \in \mathcal{J}} J = \{s^*\}$, i.e., can we find a small subset of $\mathcal{J} \subseteq \mathcal{K}$ such that its intersection contains exactly s^*. One can see that $\mathcal{K}$ contains the set of questions that can be asked about s^*. We can safely assume that $\forall K \in \mathcal{K}, s^* \in K$.

Set cover. Let $\mathcal{A}$ be an arbitrary set and $\mathcal{B} \subseteq 2^{\mathcal{A}}$. The decision problem is whether one can find some set $B^* \subseteq \mathcal{B}$ of size k or smaller such that $\bigcup B^* = \mathcal{A}$.

NP-completeness. We show that the set cover problem reduces to the set intersection problem.

Lemma C.1. *Set cover problem reduces to the Set intersection problem.*

Proof. we define $\mathcal{S} = \mathcal{A} \sqcup \{s^*\}$ for some dummy element s^* and $\mathcal{K}$ to contain the subsets that are complements of elements of $\mathcal{B}$, with s^* added, $\mathcal{K} = \{K_i = \overline{B_i} \sqcup \{s^*\} | B_i \in \mathcal{B}\}$. Take a solution $K^* \subseteq \mathcal{K}$ to this set intersection problem and the corresponding subset $B^* = \{B_i | K_i \in K^*\}$. Since $\bigcap_{i:K_i \in K^*} K_i = \{s^*\}$, then $\bigcup_{i:K_i \in K^*} \overline{K_i} = \mathcal{S} \backslash \{s^*\}$. Thus $\bigcup_{i:B_i \in B^*} B_i = \bigcup_{i:K_i \in K^*} \overline{K_i} = \mathcal{S} \backslash \{s^*\} = \mathcal{A}$. □

For completeness, we also show the other direction of reduction.

Lemma C.2. *Set intersection problem reduces to the Set cover problem.*

Proof. We define $\mathcal{A} = \mathcal{S} \setminus \{s^*\}$ and $\mathcal{B}$ contains subsets that are the complements of the elements of $\mathcal{K}$, $\mathcal{B} = \left\{B_i = \overline{K_i} \mid K_i \in \mathcal{K}\right\}$. Take a set cover $B^* \subseteq \mathcal{B}$ of $\mathcal{A}$ and its corresponding subset $K^* = \{K_i | B_i \in B^*\}$ of $\mathcal{K}$. Since B^* covers $\mathcal{A}$, i.e. $\bigcup B^* = \mathcal{A}$, we have $\bigcap_{B_i \in B^*} \overline{B_i} = \phi$, and thus K^*: $\bigcap_{i:K_i \in K^*} K_i = \{s^*\} \cup \bigcap_{B_i \in B^*} \overline{B_i} = \{s^*\}$. □

Finally, any solution to the set intersection problem can also be checked in polynomial time.

Remark C.3. A very similar problem known as *teaching dimension* was studied by [15], and is in turn closely related to other related concepts in learning theory such as the VC-dimension.

D Connections to Classical Search Problems

SLS builds upon classical combinatorial search and decision-tree optimization [28]. For a fixed question set, a deterministic Questioner strategy corresponds to a binary decision tree, where internal nodes represent questions and leaves identify items. Minimizing worst-case interaction length corresponds to minimizing the maximum tree depth. Similarly, identifying a known target using the fewest questions can be formulated as a set-cover problem where each question distinguishes the target from a subset of alternative items, and the selected questions must collectively eliminate all alternatives [11]. These connections motivate Theorem 3.6 and the even-split observation which illustrate established complexity and information-theoretic properties to provide the reader with better intuition rather than establishing new theoretical contributions.

SLS is also related to the classical Renyi–Ulam search framework [39, 43], which extends Twenty Questions by allowing the answerer to provide a bounded number of false answers. Unlike these liar-game formulations, SLS assumes truthful answers and introduces adversarial behavior through the initial selection of the hidden item, with the Questioner optimizing its worst-case expected identification cost. The central challenge addressed by GoT is therefore not the classical search formulation itself, but its practical application when questions are expressed in natural language and cannot be exhaustively enumerated.

E Discussion of Safe Subgame Search

Step 2 of GoT augments the subgame to allow the Item Chooser to "re-choose" a new distribution over s^*. At first glance, doing this rather than sticking to the original distribution chosen by the Item Chooser at the start of the game seems to give them significantly more power than in the original SLSR game, since it would be able to "switch items" depending on the questions asked in actual play. In fact, this is a simple implementation of *maxmargin resolving*, a crucial step in performing *safe* subgame search [35] in general zero-sum EFGs with imperfect information. In these games, the optimal strategy of a subgame may depend on other unreached branches of the original game and as such, we cannot solve for a strategy for this subgame in isolation. Performing this safe subgame resolving provides a performance guarantee for the Questioner's subgame strategy with respect to some blueprint strategy or state value estimate in the original game. Conversely if we were to not do this, the Questioner's subgame strategy will enjoy no such guarantees and may perform arbitrarily poorly. In practice, we see an improvement over unsafe variants of subgame search during our preliminary investigation, which is reported in Appendix G.

We give a very brief overview of subgame search and maxmargin resolving. The finer details and mathematical formulation are omitted. The main goal is to establish that GoT is implemented in the same way that maxmargin resolving is, which in turn implies safety.

Safe subgame search Subgame search (also known as subgame resolving or continual resolving) is a class of methods used to perform depth-limited search in a principled manner in imperfect information extensive form games. The idea behind subgame search is to play a *blueprint* strategy until the player enters a *subgame* (an imperfect information subgame is a forest of trees, closed under both the descendant relation and membership within augmented information sets for any player, [4]). The blueprint strategy is *typically* the solution to a coarse, abstract version of the full game. Once inside the subgame, the player *resolves* for a better solution for the subgame that was reached (and that subgame only). This is analogous to the perfect information case where one only performs search

(or refinement) of a strategy in the state that was reached in actual play, since solving the original full game is intractable.

A safe subgame solving algorithm is one that is guaranteed to perform no worse than the blueprint strategy. Naive subgame solving attempts to solve the subgame by constructing a gadget game starting with a chance node which leads to all initial states in the subgame based on the probabilities (under the blueprint and the opponent strategy when solving the blueprint) of reaching them. This however, neglects the fact that resolving just the subgame that was reached could entice the opponent to change their action, rendering these chance probabilities incorrect, leading to a worse performance than the blueprint.

Maxmargin resolving Maxmargin resolving [35] is one such approach to perform subgame solving in a more principled fashion. The idea is to augment the gadget game such that the *opponent* first chooses which infoset (of the opponent) it would like to begin from, after which, a chance node enforces the probability of reaching each of those states (belonging to the opponent's infoset), based on the blueprint strategy of the main player. Letting the opponent choose which infoset (note that we may have to add dummy infosets if the opponent does not move at the beginning of the subgame) they want to begin the subgame with ensures that the main player optimizes for the minimum *margin*. Here, the margin for each of the opponent's infoset is the difference between the value in the infoset under the blueprint strategy versus the refined strategy. If the main player optimizes the value of a particular head infoset (of the opponent) too much at the expense of performing poorly at other infosets, then the opponent would choose those infosets instead, resulting in unsafe resolving. By allowing the opponent to choose initial infosets, Maxmargin avoids this explicitly by ensuring that all of the infosets are equally improved (for the main player) after refinement.

In practice, we do not actually know the values of initial states of the subgame under the best-response of the opponent (without expanding the entire subgame to its leaves) and have to resort to approximate value functions. Note that it is not immediately clear what constitutes a good value function, since the value of a state will depend on play from both players and is complicated by imperfect information. See [23] for a more thorough discussion.

GoT as maxmargin resolving First, observe that SLSR is a relatively simple EFG in that the Answerer only moves once at the beginning, after which there are no further interactions from it. In fact, we could allow the Answerer to have perfect information. Furthermore, the proper subgames (i.e., not the full game) are simply the histories H, or equivalently, the subgame beginning at $I(H)$, which comprise all questions asked but not what s^* was chosen. Note that each proper subgame only has the Questioner taking actions. Since the Answerer has perfect information, the "head infosets" of a subgame $I(H)$ for the Answerer simply correspond to the initial states of that history H. Note that we don't actually need to explicitly construct these head infosets: since the Answerer has full information, these are "dummy" infosets that have essentially one action only, which lead to the corresponding state in the lead infoset of the Questioner. This shows that the structure of the gadget game of maxmargin is exactly the same as what we propose. An example of this construction is shown in Figure 5.

In maxmargin, the payoffs under each head infoset of the subgame are shifted by the payoffs under the blueprint (or function approximation). This ensures that the *margin* is optimized as opposed to absolute payoffs. Thankfully, no such shift is necessary in our case because the approximated values at each state are $\log_2(|S(H)|)$, i.e., the log of the number of items remaining. This value is the same for *every* state in the infoset $I(H)$, thus the value of the shift is the same over the entire subgame, which, as far as solving the subgame goes has no bearing on the refined strategy.

This same argument also extends to our choice of value function for WSLSR, $h(l) = \max_{s \in S(l)} w(s) \cdot (d(l) + \log_2(|S(l)|))$, which depends only on the set of items remaining, $S(H)$. However, if our value function was chosen to also depend on the Answerer's precise choice of s^*, then such an argument would no longer hold and we would have to perform these shifts in payoffs accordingly.

Remark E.1. In theory, maxmargin resolving only performs resolving upon entering a subgame (e.g., after $d = 3$ questions are asked). In practice, GoT performs resolving at every infoset that is encountered during actual play.

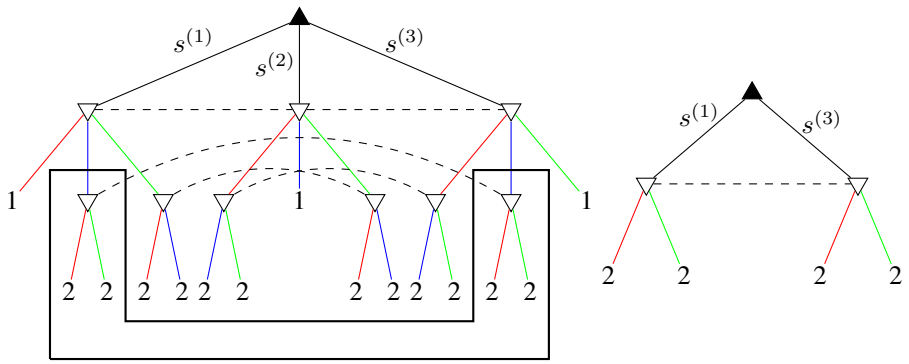

Figure 5: Left: example of a subgame based on Example 1 and Figure 1. Bounded region by thick dark lines form an example of a subgame corresponding to the history of asking $q^{(2)}$ and getting an answer of 1. Right: the gadget game used for solving the subgame shown on the left. Note that it is the Answerer who takes the first action, not chance (which would be the case for naive subgame solving), and the actions only include the items that are consistent with H, in this case $s^{(1)}$ and $s^{(3)}$.

F Implementation Details

In this section, we detail implementation details that were left out of the main paper for brevity.

F.1 General Implementation

Method Hyperparameters Since UoT shares a similar simulation procedure with GoT, we use the same values for the simulation depth d and the number of candidate questions m for both methods within each experiment, although these values may vary across different experiments. For DC, we similarly provide a set of m candidate questions which it may choose from.

Question Generation When There Are Two Items Left In instances during a (W)SLSR game where the current history H satisfies $|S(H)| = 2$, we deterministically construct two candidate questions of the form "Is x the correct item?" for each remaining item, instead of sampling questions from the LLM. While not strictly required, it is implemented primarily for efficiency. Under Assumption 3.7, any question that satisfies the assumption will successfully distinguish between the two remaining items, thereby terminating the game. This eliminates the need to query an LLM at such states, thereby reducing the latency associated with constructing the simulation tree.

Question Caching and Reuse Since we construct the SLSR game on demand in our experiments, we need to ensure the consistency between the game instance played by each of the methods. During each execution of GoT, the set of candidate questions $g(S(H))$ generated by the LLM for any explored history H is cached for reuse. In the case of UoT and DC, if the current history has previously been explored by GoT, the corresponding cached set of candidate questions is reused in place of sampling new questions from the LLM. This is done to ensure a fair comparison across methods, as variations in the quality of candidate questions can substantially impact performance.

Handling erroneous LLM outputs For ease of parsing, we require the LLMs to produce outputs in JSON format. While the model occasionally fails to adhere to this format, we address such cases by retrying the generation. In our experiments, this retry mechanism proved effective, and no further issues were observed.

F.2 Game of Thought

CFR convergence We solve each constructed subgame by running CFR for 100,000 iterations. To assess solver convergence, we measure the exploitability of the resulting strategies within the constructed subgame. Across all SLSR subgames from 10 runs on the COMMON+ dataset, LiteEFG reports exploitabilities between 0.001 and 0.005. These values are small relative to the utility scale of the game which range from 1 to 4, indicating that CFR converges to a close approximation of an

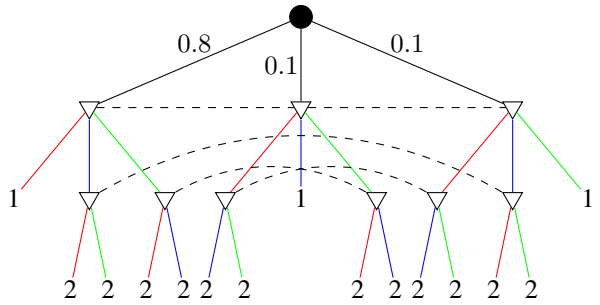

Figure 6: EFG representation of Example 4. The root node ● is a chance node and the other nodes ▽ belong to the Questioner. The respective probabilities of each item is shown.

equilibrium within each constructed subgame. We emphasize that this measures convergence of the CFR solver conditional on the constructed subgame, rather than the approximation error introduced by the subgame construction itself. In general, safe subgame solving does not provide strong guarantees with respect to the full game tree.

When Simulation Tree is Terminal During Simulation Step (1), after constructing the local simulation tree, we verify whether each leaf node corresponds to a terminal state, i.e., whether $|S(l)| = 1$ holds for all leaf nodes. If this condition is met, the strategy derived from the current subgame is adopted for all subsequent question selections until the end of the game, rather than being used solely for the next question followed by resolving a new subgame at the next step. This is justified by the fact that the current subgame extends to the end of the game, thereby obviating the need for iterative strategy construction in subsequent steps.

F.3 MISQ

Method Overview Similar to UoT and GoT, Monte Carlo Tree Search for Information Seeking Questions (MISQ) and MISQ with Hierarchical Feedback (MISQ-HF) [7] use LLMs to generate multiple candidate questions for progressively narrowing the search space. Unlike UoT and GoT however, MISQ employs MCTS to explore promising branches of the decision tree without *exhaustively* expanding all candidates, while optimizing both immediate and long-term information gain using a reward function similar to that of UoT. In our study, we follow the implementation provided by [7], keeping the hyperparameters faithful to the original implementation and providing them as default values in the supplementary code. Although [7] also propose MISQ-HF, which augments MISQ with a feedback mechanism, this mechanism is setting-dependent because it requires an initial problem description specific to each instance. Since such descriptions are not available in our datasets, we do not directly compare against MISQ-HF.

F.4 Best Responses

Obtaining Best Response To obtain a best response (BR), we consider a non-adversarial setting in which the item (distribution) is drawn from a known prior P^o rather than chosen adversarially. Under this assumption, the interaction can be modeled as a single-player extensive-form game with imperfect information, analogous to our SLS representation, by replacing the root node corresponding to the Item Chooser with a chance node whose outgoing edges are weighted according to P^o. The optimal Questioner strategy in this case is deterministic and should minimize the expected cost incurred. We obtain this strategy by adopting an iterative, on demand framework similar to GoT, replacing step 3 with a simple backward induction.

Example 4. *Consider a game similar to Example 1, with $s^{(1)}, s^{(2)}, s^{(3)}$ being distributed under the prior $P^o = (0.8, 0.1, 0.1)$.*

In Example 4, the Questioner should always ask $q^{(1)}$ first, followed by either $q^{(2)}$ or $q^{(3)}$. This yields an expected cost of 1.2. Figure 6 illustrates the extensive form representation of Example 4.

Note that in this setting, the problem is equivalent to planning under partial observability, i.e., a Partially Observable Markov Decision Process (POMDP) [42].

Method	Common	S128	Breeds
GPT 4.1			
GoT	<u>10.2</u>	<u>11.8</u>	<u>7.4</u>
Unsafe GoT	14.6	12.6	9.8
UoT	11	13	9

Table 3: **Worst-case interaction length of Safe and Unsafe variants of GoT**. Lower is better. UoT's performance is included for comparison. Best performance for each setting is <u>underlined</u>.

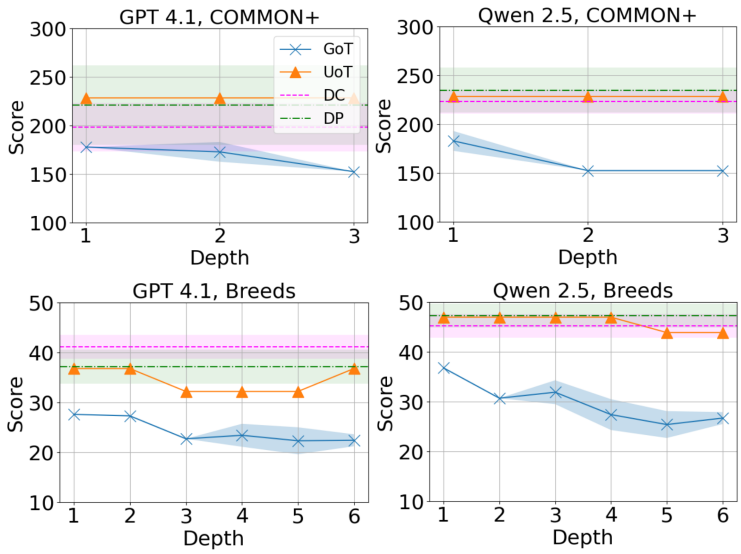

Figure 7: **Worst-case performance of various methods in Weighted Variant on Common and Breeds.** Lower is better. The x-axis is d, the y-axis is the payoff for the Item Chooser $w_i \cdot |H|$.

G Additional Results

G.1 Unsafe GoT

During preliminary stages of this project, we experimented with an unsafe version of our current GoT framework, which we will refer to as Unsafe GoT. It is mostly similar to our current framework, except in step 2 when we construct the EFG tree. In the first iteration of Unsafe GoT, we construct the EFG tree with the item chooser at the root and solve the game as per normal GoT. In the subsequent iterations, we use a chance node with the marginal probabilities obtained based on the item chooser strategy at the first iteration for the remaining set of consistent items. This essentially stems from the assumption that the Item Chooser would maintain its original strategy regardless of how the Questioner chooses its questions in the subsequent iteration. The results of using the unsafe variant are shown in Table 3. As mentioned, the unsafe algorithm can perform arbitrarily poorly, highlighting the importance of safe subgame resolving.

G.2 Effect of Simulation Depth

We further study the effect of increasing the simulation depth d for both UoT and GoT; the results are shown in Figure 7. Overall, GoT improves as d increases, but the gains plateau as it approaches what we hypothesize to be the optimal strategy for the fixed set of LLM-sampled questions. In contrast, UoT's worst-case performance shows little to no improvement even with deeper lookahead, indicating that optimizing for information gain does not necessarily translate into better worst-case guarantees.

G.3 SLSR with Synthetic Splits

The quality of a candidate question q in SLSR at some infoset $I(H)$ can be roughly measured using the ratio $|Y(S(H), q)| : |\bar{Y}(S(H), q)|$, representing how evenly q is able to split $S(H)$ into two.

Method	r 0.4	0.33	0.25
3 Features			
GoT	9.4	10.0	13.8
UoT	10.0	11.0	15.0
5 Features			
GoT	9.2	10.2	13.2
UoT	10.0	11.0	15.0
Random Splits			
GoT	8.8	9.8	11.6
UoT	10.0	11.0	15.0

Table 4: **Worst-case interaction length on SLSR with Synthetic splits with Common**. Lower is better. d is set to be 3.

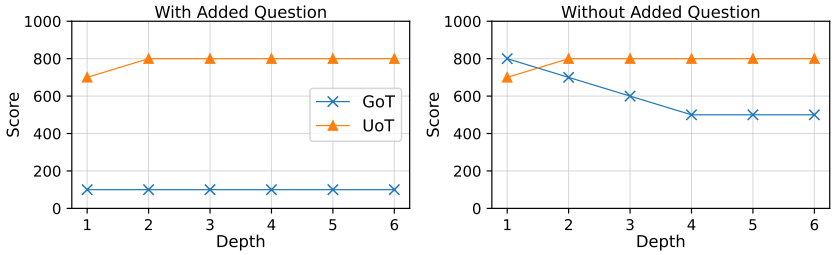

Figure 8: **Performance in the weighted variant with artificially skewed weights on Breeds.** Lower is better. Graph on the left shows the performance when the additional question is included.

Given the difficulty in controlling the quality of LLM-sampled questions, we wish to study SLSR in a more controlled environment. Instead of using an LLM for question generation, we define a question generating function $g' : 2^{\mathcal{S}} \setminus \emptyset \times (0,1) \to 2^{\mathcal{S}}$ which takes as input $S(H)$ and a split ratio $r \in (0,1)$, and outputs the set[3] $Y' \subset S(H)$ (and consequently $\bar{Y}' = S(H) \setminus Y'$) at the fixed ratio $\frac{|Y'|}{|S(H)|} = r$. Two implementations of g' were considered: (i) Random Splits: Sample and return $r \cdot |S(H)|$ items uniformly at random from $S(H)$ without replacement. (ii) Feature Based Splits: For each $s \in S(H)$, generate k features $F_i = (f_1, ..., f_k)$, $f_j \in [0,1]$ at the start of the game. g' randomly selects one of k features, and sorts $S(H)$ based on the selected feature. The top $r \cdot |S(H)|$ items are returned.

The results are shown in Table 4. GoT consistently achieves superior performance compared to UoT, particularly as r decreases and the splits become increasingly skewed. This suggests that GoT provides the most improvement when the candidate questions are not optimal. This however does not translate well when actual questions are used due to other confounding factors not captured.

G.4 Question Quality

We know that if an LLM is always able to propose questions that perfectly split $S(H)$ into even halves, we would be able to achieve the optimal performance by always asking such a question. However, we see that this is not the case: all methods fall short of the theoretical optimal performance of $\log_2(|\mathcal{S}|)$ by around 2 to 3 turns of interaction. This suggests that in practice, particularly in realistic settings such as medical diagnosis or troubleshooting, it may be difficult to phrase a sufficiently natural question that achieves an even split. In such cases, considering multiple candidate questions and adopting a randomized selection strategy, as GoT does, can hedge against outliers that require an excessively long conversation.

We further conducted a brief ablation to examine the impact of question quality in WSLSR. Specifically, we created an artificially skewed weight distribution on the Breeds dataset by assigning weight 1 to all but one item and weight 100 to the remaining item. Under this construction, the optimal first query is to test whether the highest-weight item is the target. We verify this by manually injecting the corresponding question into the initial candidate set. As shown in Figure 8, GoT consistently

[3]The question itself does not matter in this case, as both UoT and GoT are language agnostic.

Method	Experiment	
	Subgame ($d = 3$)	**Full Game ($d = D$)**
GPT 4.1 + Even split prompt		
GoT	6.4	5.84
UoT	7.0	7.0
GPT 4.1 + Natural prompt		
GoT	7.4	5.64
UoT	9.0	8.0
Qwen 72B Instruct + Even split prompt		
GoT	6.2	5.44
UoT	7.0	7.0
Qwen 72B Instruct + Natural prompt		
GoT	6.6	5.88
UoT	8.0	8.0

Table 5: **Worst-case interaction length for full SLSR Game.** Lower is better. Dataset used is the Breeds Dataset.

reaches this optimal strategy. In contrast, when this question is not included, performance degrades substantially, indicating that GoT's effectiveness can be sensitive to the quality of the candidate questions.

G.5 Full SLSR Games

For each SLSR game, there exists a lower bound on the expected worst- case number of questions required. This bound can be obtained by solving the fully specified game. As this is only feasible for smaller datasets, we constructed complete SLSR games on the smaller Breeds dataset and experimented on GoT and UoT by setting the simulation depth d to game tree depth D. Since constructing the full game tree is required, we enforce Assumption H.1 to ensure that all branches eventually terminate. The results are shown in Table 5, where GoT's performance aligns with the lower bound on the performance. On the other hand, UoT's performance showed minimal improvement even when provided with the full game tree during simulation, suggesting that achieving the lower bound requires a non-deterministic strategy.

G.6 Item Weights

Figure 9 presents a histogram of item weights used in the weighted variant for the Breeds and Common datasets in the 20Q setting, DX in the MD setting, and FD in the TS setting. For datasets in the 20Q setting, the weights were assigned randomly by sampling from a lognormal distribution with parameters $\mu = 0, \sigma = 1$, as there was no clear method for determining meaningful valuations for each item given the nature of the datasets. For datasets in the MD and TS settings, the weights are obtained by asking Chatgpt 5.2 to annotate each disease/fault with a severity score. The prompt used is shown in Table 6.

It is important to note that the performance of GoT in a WSLSR game may be sensitive to the underlying distribution of item weights. In particular, when the weights were sampled from a uniform distribution, we observed that the relative performance of GoT compared to UoT mirrored the results in the unweighted setting, providing a consistent but smaller improvement.

G.7 Randomization of GoT's strategies

We quantified the randomness of GoT's strategy using entropy, as shown in Figure 10. In general, the resulting strategies exhibit higher entropy in the standard SLSR game compared to the WSLSR variant. This is likely due to the item weight distribution being tail-heavy, which encourages the strategy to favor candidate questions that more efficiently isolate the heavily weighted items. Nonetheless the strategies remain sufficiently random, suggesting that we do not encounter degenerate cases in which a deterministic strategy would be optimal.

MD
Here is a list of diseases. I want you to assign a severity score between 1 to 10 for each of these diseases. This severity score should depend on how life threatening the disease can be, and how urgently it needs treatment or care. The list of diseases: **[diseases]**
TS
Here is a list of car faults. I want you to assign a severity score between 1 to 10 for each of these faults. This severity score should depend on how severe the fault can be, and how urgently it needs to be repaired. The list of car faults: **[faults]**

Table 6: **Prompt used to annotate the item weights for the MD and TS settings.**

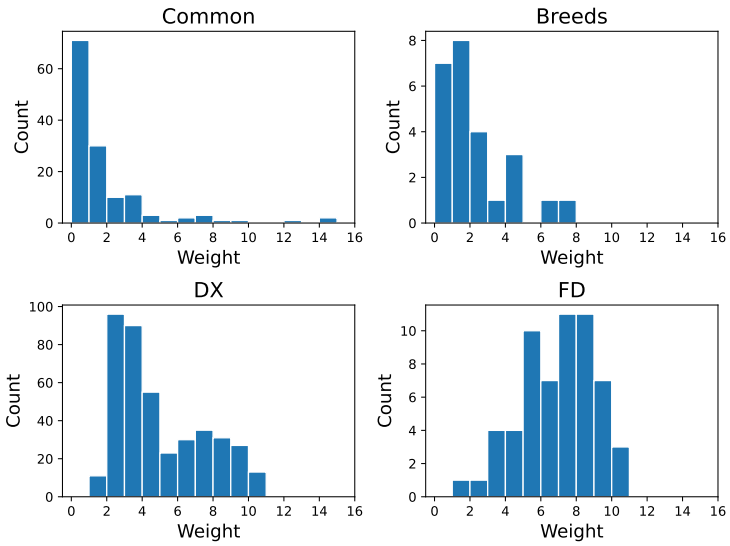

Figure 9: **Distribution of item weights used in our experiments.**

G.8 Token Statistics

As seen in Table 7, GoT and UoT have comparable computational complexity in terms of token count and total LLM queries, as both are dominated by the same simulation rollout step. Thus, GoT can be viewed as a robust extension of UoT that provides stronger worst-case performance with little additional computational overhead. Although pure-prompting baselines such as DP and DC are cheaper, they are less effective at complex strategic decision-making, as reflected by their weaker performance in Tables 1 and 2.

H Discussion on Assumptions

All of the aforementioned assumptions are either naturally satisfied by the experimental setup or are explicitly enforced during the execution of our method.

Assumption 3.1 In our formulation of the SLS problem, we assume that (i) the codomain of f is binary, and (ii) each item $s \in \mathcal{S}$ has a unique, well-defined answer under f. However, this assumption may not hold in real-world scenarios due to the inherent ambiguities and interpretive variability commonly present in natural language. One source of such ambiguity arises when an item name

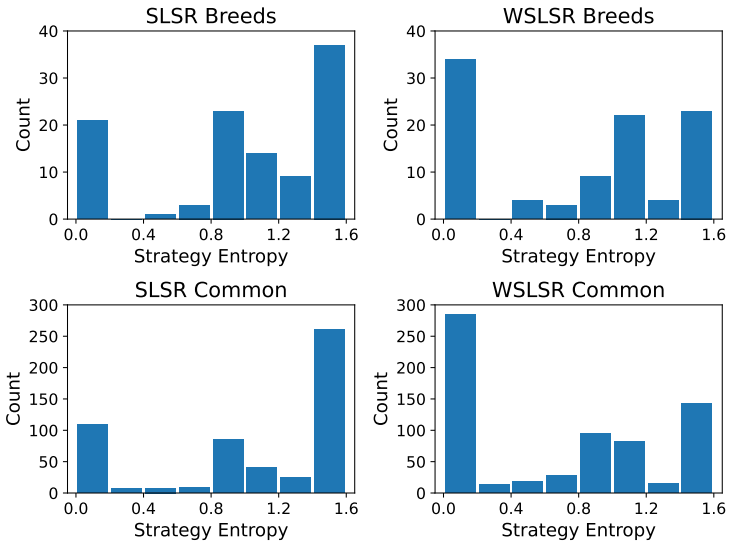

Figure 10: **Entropy of the strategies obtained using GoT at each infoset.** As $m = 3$, the maximum entropy of the strategy is $\log_2(3) \approx 1.58$ when every candidate question is chosen with equal probability. Strategies for when there are only two items left are not included.

Method	Token Usage
GoT	18960
UoT	18738
MISQ	6432
DP	855
DC	1442

Table 7: **Average token usage per turn of interaction.** Dataset used is the DX Dataset. For both UoT and GoT, the $d = m = 3$. Model used is GPT 4.1

refers to multiple distinct entities. For example, consider the question: Is the Titanic something a person can pick up with their hands? The answer would be no if referring to the actual ship, but yes if referring to a DVD of the movie Titanic. In our experiments, to abstract away such linguistic ambiguity, we instantiate the labeling function f with an LLM and treat its output as a definitive label for each item.

Assumption 3.2 This naturally follows from Assumptions 3.1 and 3.9 and the game structure.

Assumption 3.3 This naturally follows from the game structure.

Assumption 3.4 We employed LLMs, accessed both via online APIs and locally hosted instances, to implement the function f. For online API-based queries, the average latency per query is approximately 1 to 5 seconds, while for locally hosted models, the latency is slightly lower, typically around 1 to 3 seconds.

Assumption 3.5 This assumption is satisfied as $\mathcal{Q}$ is the set of all possible natural language questions. In particular, it is trivially satisfied by the presence of identity questions.

Assumption 3.7 This assumption holds in nearly all cases when LLMs are used as the question generator g. To enforce this, we verify whether there exists at least one question from the sampled candidate set that permits progression of the game; if no such question is found, we resample the candidate questions. For certain experiments (such as full SLSR game solving) we impose a stronger version of this assumption:

Assumption H.1. For every subset $S \subseteq \mathcal{S}$, for every $q \in g(S)$, there exists a pair of distinct items $s, s' \in S$ where $f(q, s) \neq f(q, s')$

This guarantees that the game tree will be finite, thereby enabling the complete construction of the game tree. To enforce this condition, we explicitly remove any candidate question that fails to satisfy the assumption from the set of candidate questions.

Assumption 3.9 This assumption is made to mitigate potential inaccuracies arising from LLM-based classifications. Although LLMs may produce erroneous outputs, handling such errors is beyond the scope of our method and is therefore not the primary focus of this work. Without this assumption, classification errors during the simulation process could, in the worst case, require a restart of the game. Specifically, if the target item s^* is misclassified, it may be incorrectly eliminated from the set of possible items upon receiving a contradictory response from the answerer. Moreover, we observed that such misclassifications and consequently restarts occurred frequently when attempting to reproduce the results of [20], which drastically affected the worst-case performance. Since this does not influence the Questioner's strategy for any of the evaluated methods, we believe it does not compromise the fairness of the experimental results. In real life deployment, one can employ multiple oracles and use the aggregated answer to reduce the likelihood of such misclassifications, and fall back to restarting the interactions if it does occur. To enforce this assumption for both GoT and UoT, we depart from the approach used in [20], where a separate LLM instance was employed to simulate the Answerer. Instead, we cache the item set splits generated during the simulation step corresponding to each question and reuse them as the Answerer's response during our experiments. This strategy ensures consistency between the item classifications performed during the simulation phase and the responses observed during the actual execution of each method.

I Dataset Details

All constructed datasets (along with item weights used for the weighted variant) are released alongside the accompanying code.

20Q To construct the datasets, we relied on publicly available sources. For example, the American Kennel Club for the Breeds dataset, and ChatGPT for the S128 dataset. All AI-generated datasets were subsequently verified to ensure that each item included corresponds to a real-world entity. The size of these datasets is intentionally designed to be comparable to those used in [20], since both their method (UoT) and GoT share a similar computational complexity due to the future simulation steps. Game-solving in GoT contributes only a minor fraction to the running time, and GoT can be seen as a robust extension to UoT at little extra cost. This is also why we primarily compare our method against UoT.

Medical Diagnosis (MD) The *DXBench(DX)* dataset [6] consists of 1148 entries of real cases across 461 unique diseases. We sampled 100 out of the 461 unique diseases uniformly at random to be used as the item hypothesis space. For the average case analysis, we count how many entries in the DX dataset have each of the 100 diseases as the final diagnosis, and use this as an estimate of the population distribution.

Troubleshooting (TS) *FlowDial(FD)* [38] contains both laptop-related and car-related faults. We decided to focus on the latter due to its greater variety and clearer distinction between each fault. In the dataset, each fault corresponds to a short textual description at a leaf node of a directed graph that encodes a troubleshooting flowchart. We extract these leaf descriptions and prompt GPT-5.2 Thinking to map each one to a concrete car-fault label. We then manually verify and edit the resulting labels based on the associated troubleshooting dialogues provided in FD. Finally, we merge or remove near-duplicate faults (e.g., "water leakage" vs. "water pump leak") to ensure that each retained fault is distinct and readily distinguishable.

Item Weights We assign weights to each item in the weighted variant of the problem. For MD and TS, we use GPT-5.2 Thinking with extended reasoning to assign each item an integer weight from 1 to 10, reflecting the severity and urgency of the disease or fault. Since 20Q lacks an analogous notion of severity, we instead sample weights from a lognormal distribution with $\mu = 0$ and $\sigma = 1$.

Method	20Q		
	Common	S128	Breeds
GPT 4.1 + Even split prompt			
GoT	9.4	9.2	6.4
UoT	10.0	10.0	7.0
DP	11.7	12.9	7.8
DC	12.3	12.6	9.0
GPT 4.1 + Natural prompt			
GoT	10.2	11.8	7.4
UoT	11.0	13.0	9.0
DP	13.8	16.2	7.8
DC	12.9	14.6	9.3
Qwen 2.5 72B Instruct + Even split prompt			
GoT	10.2	10.2	6.2
UoT	11.0	11.0	7.0
DP	11.9	14.6	8.0
DC	12.4	13.6	8.3
Qwen 2.5 72B Instruct + Natural prompt			
GoT	10.0	10.8	6.6
UoT	11.0	12.0	8.0
DP	12.7	19.2	8.0
DC	12.7	17.9	7.8

Table 8: **Worst-case interaction length for each method using even or natural prompts**. Lower is better. For both UoT and GoT we set $d = 3$.

J Examples of Questions Asked

Natural Questions A typical question posed by the LLM in a game of Twenty Questions with the Common dataset resembles the following:

- Is the item something a person can physically pick up with their hands (assuming average human size and strength)?
- Is the item man-made?
- Is the item an animal?

Such questions generally pertain to intrinsic properties of the items themselves.

In contrast, when the *even* prompt was used, we occasionally observed the generation of more "unnatural" questions:

- Is the breed's name made up of only one word?
- Is the name of the item longer than 7 characters?
- Does the item's name start with a letter from A-M (inclusive)?

While we find it difficult to provide a quantitative evaluation of the "naturalness" of a question, intuitively, unnatural questions typically reference the item names directly, rather than the underlying items themselves. Natural questions on the other hand generally focus on intrinsic properties of the items. Unnatural questions can be interpreted as the LLM's attempt to optimize the question by improving the chances of obtaining an even-split, which is optimal. These occurrences were primarily observed when GPT-4.1 was used as the LLM. We suspect this to be why a noticeable improvement in DP can be observed when *even* is used over *natural*, as seen from Table 8. This is less noticeable in GoT and UoT, possibly due to both performing planning via lookahead. This could reduce the influence of the few "unnatural" questions on the overall performance.

Questions in Practical Settings As noted previously, we claimed that the questions generated by the LLM in the MD and TS settings are reasonably natural. To illustrate this, we provide representative examples of questions sampled from GPT 4.1 in each setting.

Examples of questions sampled for Medical Diagnosis:

- Is the main symptom dermatological, affecting the skin (e.g., rashes, eczema, pigmentation, lesions)?
- Is the disease primarily associated with an infection or inflammation (for example, hepatitis, bronchitis, pneumonia, infectious mononucleosis, sepsis, pharyngitis, gastritis)?
- Is the primary symptom related to the respiratory system (such as cough, shortness of breath, or chest pain)?

These questions target either the patient's symptoms or the affected body systems, which aligns with how diagnosis is typically conducted in practice. Moreover, each question can be (i) posed to the patient with minor rephrasing, (ii) used as a mental checklist during clinical reasoning, or (iii) mapped to a diagnostic test relevant to specific diseases.

Examples of questions sampled for Troubleshooting:

- Do you notice any warning lights or indicators on the dashboard when you turn the key to the 'ON' position?
- Are there any electrical accessories (like headlights or dashboard lights) not functioning even when the ignition is off?
- Does the car fail to start or crank when turning the ignition key?

These questions are easy to understand and, importantly, refer to symptoms that can be readily verified by the customer. This property makes them suitable for the context of troubleshooting.

Ambiguous Questions In our formulation of the SLS problem, we assume that (i) the codomain of f is binary, and (ii) each item $s \in \mathcal{S}$ has a unique, well-defined answer under f, as stated in Assumption 3.1. However, this assumption may not hold in real-world scenarios due to the inherent ambiguities and interpretive variability commonly present in natural language. One source of such ambiguity arises when an item name refers to multiple distinct entities. For example, consider the question: Is the Titanic something a person can pick up with their hands? The answer would be no if referring to the actual ship, but yes if referring to a DVD of the movie Titanic. While language ambiguity is a well-studied phenomenon in the field of Natural Language Processing, addressing such ambiguities remains challenging when applied to the more formal and rigorous framework of Game Theory. We believe that addressing this challenge presents a promising direction for future research.

K Prompts Used

The prompts used for sampling questions (to implement the function g) in the weighted and unweighted variants are shown in Tables 9 and 10 respectively. DP uses the same prompts for asking questions by setting $m = 1$. DC additionally uses the prompt in Table 11 to choose the question from the candidate set. The prompts used to generate answers to questions for items (to implement the function f) are shown in Table 12.

L Experiment Hardware

Running the EFG solver was done on an AMD Ryzen 5 5500 CPU with 32 GB RAM. Hosting the local instance of Qwen 2.5 72B Instruct LLM was done via the vLLM [26] library on either 2 Nvidia H200 GPUs each with 141 GB memory, or 4 Nvidia H100 GPUs each with 80 GB memory. Each run of the experiment on the COMMON dataset took around 6 hours.

M Broader Impacts

The goal of GoT is to advance machine learning methodology, and the framework may be applicable across multiple domains. However, its use in high-risk settings should be approached with care. In particular, applications to sensitive domains such as medical diagnosis may carry risks, including misdiagnosis. Although GoT shows promising results in such settings, we emphasize that we have

Even Prompt 20Q
We are playing the game of ask and seek, where I am the answerer and you are the questioner. I have chosen one item from a list of items, and your goal is to identify the correct item from a list of items by asking a series of yes or no questions. You should try to ask as few questions as possible. Here are all the items: **[items]** One of the items is the correct item, please design questions about those items that helps you find the right item and can only be answered by Yes or No. *Notably, this question should fulfill that the count of YES items and NO items are almost the same with a permissible discrepancy of no more than one!* You should respond with the best **[m]** questions you can think of.
Natural Prompt 20Q
We are playing the game of ask and seek, where I am the answerer and you are the questioner. I have chosen one item from a list of items, and your goal is to identify the correct item from a list of items by asking a series of yes or no questions. You should try to ask as few questions as possible. Here are all the items: **[items]** One of the items is the correct item, please design questions about those items that helps you find the right item and can only be answered by Yes or No. *You should avoid asking unnatural questions such as ones about the number of words the item has, or any such similar questions.* You should respond with the best **[m]** questions you can think of.
MD
You are a doctor trying to diagnose a patient. The patient suffers from one of the diseases from the provided list, and your goal is to identify the correct item from this list asking a series of yes or no questions. You should aim to ask as few questions as possible. Here are all the diseases: **[diseases]** One of the diseases is the correct disease, please design questions about those items that help you find the right item and can only be answered by Yes or No. You should respond with the best **[m]** questions you can think of.
TS
You are a car mechanic providing remote troubleshooting to a customer. The customer's car suffers from one of the faults from the provided list, and your goal is to identify the correct fault from this list asking a series of yes or no questions. You should aim to ask as few questions as possible. Here are all the faults: **[faults]** One of the faults is the correct fault, please design questions about these faults that help you find the right item and can only be answered by Yes or No. You should respond with the best **[m]** questions you can think of.

Table 9: **Prompts used to sample candidate questions for unweighted variant.** The main difference between even and natural prompts are *italicized*. The items are represented as a Python list, e.g., ['Oppenheimer', 'Alan Turing', 'A Beautiful Mind'], while m is an integer

not conducted clinical trials or validated the framework for clinical use. Therefore, GoT should not be used to provide medical advice.

20Q
We are playing the game of ask and seek, where I am the answerer and you are the questioner. I have chosen one item from a list of items, and your goal is to identify the correct item from a list of items by asking a series of yes or no questions. Each item has a value associated with it, and your penalty score will be the number of questions asked until you find the right item multiplied by the value of the correct item. You should try to minimize this score. Here are all the items and their values: **[items with weights]** One of the items is the correct item, please design questions about those items that helps you find the right item and can only be answered by Yes or No. You should avoid asking unnatural questions such as ones about the number of words the item has, or any such similar questions. You should respond with the best **[m]** questions you can think of.
MD
You are a doctor trying to diagnose a patient. The patient suffers from one of the diseases from the provided list, and your goal is to identify the correct item from this list asking a series of yes or no questions. Each disease has a severity associated with it. Your penalty will be the number of questions asked multiplied by the severity value of the correct disease. You should try to minimize this penalty. Here are all the diseases: **[diseases with weights]** One of the diseases is the correct disease, please design questions about those items that help you find the right item and can only be answered by Yes or No. You should respond with the best **[m]** questions you can think of.
TS
You are a car mechanic providing remote troubleshooting to a customer. The customer's car suffers from one of the faults from the provided list, and your goal is to identify the correct fault from this list asking a series of yes or no questions. Each fault has a severity associated with it. Your penalty will be the number of questions asked multiplied by the severity value of the correct fault. You should try to minimize this penalty. Here are all the faults: **[faults with weights]** One of the faults is the correct fault, please design questions about these faults that help you find the right item and can only be answered by Yes or No. You should respond with the best **[m]** questions you can think of.

Table 10: **Prompt used to sample candidate questions for weighted variant.** Items with weights are represented as a Python dictionary, e.g., {'Oppenheimer' : 3, 'Alan Turing' : 2, 'A Beautiful Mind' : 2}, while m is an integer

Choosing Prompt for 20Q Unweighted Variant

We are playing the game of ask and seek, where I am the answerer and you are the questioner.
I have chosen one item from a list of items, and your goal is to identify the correct item from a list of items by asking a series of yes or no questions.
You should try to ask as few questions as possible.
Here are all the items:
[items]
Here are some Yes or No question about them:
[question]
One of the items is the correct item, please choose one question from the list of questions that helps you find the right item.

Choosing Prompt for 20Q Weighted Variant

We are playing the game of ask and seek, where I am the answerer and you are the questioner.
I have chosen one item from a list of items, and your goal is to identify the correct item from a list of items by asking a series of yes or no questions.
Each item has a value associated with it, and your penalty score will be the number of questions asked until you find the right item multiplied by the value of the correct item. You should try to minimize this score.
Here are all the items:
[items with weights]
Here are some Yes or No question about them:
[question]
One of the items is the correct item, please choose one question from the list of questions that helps you find the right item.

Choosing Prompt for MD Unweighted Variant

'You are a doctor trying to diagnose a patient.
The patient suffers from one of the diseases from the provided list, and your goal is to identify the correct item from this list asking a series of yes or no questions.
You should aim to ask as few questions as possible.
Here are all the diseases: **[diseases]**
Here are some Yes or No question about them: **[question]**
One of the diseases is the correct disease, please choose one question from the list of questions that helps you narrow down to the right disease as quickly as possible.

Choosing Prompt for TS Unweighted Variant

You are a car mechanic providing remote troubleshooting to a customer.
The customer's car suffers from one of the faults from the provided list, and your goal is to identify the correct fault from this list asking a series of yes or no questions.
You should aim to ask as few questions as possible.
Here are all the faults: **[faults]**
Here are some Yes or No question about them: **[question]**
One of the faults is the correct fault, please choose one question from the list of questions that helps you narrow down to the right fault as quickly as possible.

Table 11: **Prompts used by DC to choose questions.** Questions are represented as a Python dictionary, e.g., {0 : 'Related to codes?', 1 : 'Is it a movie?', 2 : 'Is it a person?'}. The weighted variants of the prompt for MD and TS setting is obtained by modifying the unweighted variant in a similar manner as in the 20Q setting.

Answer Prompt

Here are some **[items/diseases/car faults]**:
[items]
Here is a Yes or No question about them:
[question]
Please classify the **[items/diseases/car faults]** above based on this question. You should ensure that each item should only appear once in the final classification. If you are unsure about any item, you can mention it in the elaboration.

Table 12: **Prompt used to get answers to questions.** We refer to the hypothesis space as items, diseases or car faults for the 20Q, MD, and TS setting respectively.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Key contributions highlighted at the end of Section 1.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Discussed in Section 7 and Appendix A and H

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Proofs in C

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Details in Section 6 and Appendix F

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code and data provided in submission

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Details in Section 6 and Appendix F

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: t-test for main results included in Appendix G

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Hardware and compute time stated in L

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.

- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research presented in this paper adheres to all the guidelines specified in the NeurIPS Code of Ethics. We have ensured that our methods and experiments are conducted responsibly and ethically.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Discussed in Appendix Section M

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not see any such risks involved in our work.

    Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All used data and code are properly cited.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Code is submitted in the supplementary material.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

Justification: There are no crowdsourcing or human subjects involved in this work.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: There are no crowdsourcing or human subjects involved in this work.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLMs are part of the proposed framework as described in Section 5

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.