Active Task Disambiguation with LLMs

Kobalczyk, Astorga, Liu & van der Schaar · ICLR 2025 · University of Cambridge

When ambiguity is the norm — how should LLMs ask better questions?

CS 8803-LLM · Paper Presentation

The Problem: Ambiguity Is Everywhere

LLMs are evaluated on clean benchmarks — but real users are not clean

Write a function to search for an item in a list.

One prompt. Four valid but completely different solutions.

```
return x in lst
```

Returns boolean

```
return lst.index(x)
```

Returns index, crashes if missing

```
return lst.index(x) if x in lst else None
```

Returns index or None

```
return [i for i,v in enumerate(lst) if v==x]
```

Returns ALL indices

Why This Is More Than a UX Problem

Safety-critical stakes

In medical diagnosis or treatment planning, generating a response that doesn't match the user's true intent can cause real harm. Guessing wrong isn't just annoying — it's dangerous.

RLHF only covers average users

Preference optimization (RLHF, DPO) aligns LLMs to the average population. If your specific intent deviates from the average, the model will likely miss your needs — even after alignment.

Hallucinations compound ambiguity

Even if the problem statement is technically unambiguous, LLMs may still generate incorrect outputs. Task elicitation helps because example inputs/outputs positively bias the model toward correct solutions.

No ground truth for intent

Unlike factual questions, there's no database to look up 'what does this user actually want.' The LLM must infer it. Asking clarifying questions is the only principled way to narrow that space.

Two Ways to Handle Ambiguity

Preference Optimization

(RLHF / DPO)

Align the model to average user preferences.

Works when: user intent matches the population average.

Fails when: individual user deviates from average.

Requires: expensive fine-tuning on human labels.

Scales: poorly to niche or novel domains.

Iterative Task Elicitation

← This paper

Ask clarifying questions to narrow the solution space.

Works when: user can answer targeted questions.

Handles: any individual user's specific needs.

Requires: smart question selection strategy.

Scales: naturally — no fine-tuning needed.

Formalizing Task Ambiguity

Making the intuition mathematically precise

$S = (R, C)$	Problem statement. R = requirements (objective, checkable). C = context (subjective, preference-based).
$H = \{h : h \vdash R\}$	All solutions that satisfy the requirements. This is the compatible solution set.
$H^* \subset H$	The true set of solutions the user actually wants. The user's real intent.
Ambiguity	S is ambiguous when $H \supset H^*$ — there are more solutions that technically fit the requirements than the user actually wants.

Goal: find questions that shrink H toward H* with minimum questions asked.

What Makes a Question Informative?

The key insight from Bayesian Experimental Design

A good question splits the solution space into equal halves.

Think: 20 Questions game. 'Is it bigger than a cat?' halves the animal kingdom. 'Is it green?' does not.

BEST

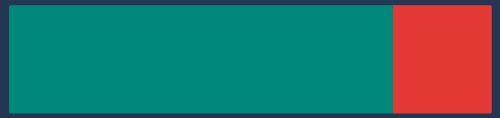
Does it return a boolean?

50% Yes

50% No

POOR

Does it handle edge cases?

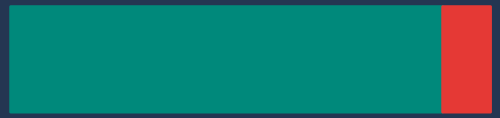


80% Yes

20% No

POOR

Is it a recursive function?



90% Yes

10% No

Expected Information Gain — The Math

Plain language walkthrough of the key formula

$$EIG(q) = H[\, p^*(h|S) \,] - E[\, H[\, p^*(h \mid S \cup (q,a)) \,] \,]$$

Expected Information Gain = Current uncertainty − Expected remaining uncertainty after asking q

$H[p^*(h\|S)]$	Shannon entropy of the current solution distribution. How uncertain are we right now? More possible solutions = higher entropy = more uncertainty.
$S \cup (q, a)$	The updated problem statement after getting answer a to question q. Each answer extends the requirements, shrinking the compatible solution set.
$E[\, H[p^*(h \mid S \cup (q,a))] \,]$	Expected remaining uncertainty after asking q — averaged over all possible answers. We don't know the answer yet, so we take the expectation.
Uniformity assumption	Since p* is unknown, the paper assumes uniform distribution over compatible solutions. This makes EIG computable: it becomes the entropy of the partition sizes.

The ATD Method — How It Works

Shift reasoning from question space to solution space

1 Sample solutions

Generate N diverse sample solutions from the current problem statement. These approximate the compatible solution set H.

2 Sample questions

Generate M candidate clarifying questions from the current problem statement.

3 Get pseudo-answers

For each question, evaluate it against each sample solution. What would the answer be if this solution were correct?

4 Estimate EIG

Compute entropy of answer distribution for each question. The question that produces the most balanced split has highest EIG.

5 Ask best question

Present the highest-EIG question to the real user. Get their actual answer.

6 Update & repeat

Extend the problem statement with the new requirement. Repeat until the solution space is small enough to commit.

Why Explicit Reasoning Works Better

The fundamental hypothesis of the paper

LLMs are better at generating solutions than at reasoning about which question to ask.

Asking a good clarifying question requires meta-cognitive reasoning — thinking about your own uncertainty. LLMs have limited exposure to this in their training data. But generating diverse solutions? LLMs are excellent at that.

Implicit reasoning (baseline)

Prompt: 'Ask the best clarifying question.'

LLM must somehow reason about what it doesn't know, what the user might mean, and what question would be most discriminative — all in one implicit forward pass.

This is hard. The training data barely covers it.

Explicit reasoning (ATD)

Generate solutions (LLM is good at this).
Generate questions (easier than selecting).
Evaluate questions against solutions (concrete).
Select by EIG (algorithmic, not LLM).

Bootstraps weak skills using strong skills.

Computing EIG in Practice

How the algorithm estimates information gain from a sample

Algorithm: estimate_EIG(q, {answers})

1. Collect all N pseudo-answers
   $\{a_1, a_2, ..., a_N\}$ from N sample solutions

2. Find unique answer categories
   e.g., {Yes, No} or {TypeError, None, Index}

3. Count how many solutions gave each answer
   n_i = count of solutions answering 'a_i'

4. Compute proportions
   $p_i = n_i / N$

5. Return entropy
   $EIG \approx -\Sigma\, p_i \log(p_i)$

Highest entropy = most balanced split = best question.

Concrete example

N = 4 solutions
M = 1 question:
'Does it return a boolean?'

Solution 1: Yes
Solution 2: No
Solution 3: No
Solution 4: Yes

Counts: Yes=2, No=2
Proportions: p=0.5, p=0.5

Entropy = -(0.5·log0.5
+ 0.5·log0.5)
= 1.0 ← maximum!

This is the best possible split.

Experiment 1: The 20 Questions Game

A clean testbed for evaluating question-asking strategies

Setup:

One player thinks of a secret animal. The LLM agent asks yes/no questions to guess it. 10 rounds. 15 animals. 5 seeds each. GPT-3.5-turbo and GPT-4o-mini.

Four strategies compared:

`implicit`	Just prompt: 'Ask the best yes/no question.' One question generated.
`implicit-ToT`	Generate M=5 questions, then prompt the LLM to pick the best one from the list.
`EIG-logprobs`	EIG estimated using token log-probabilities from the LLM's generative distribution.
`EIG-uniform`	EIG estimated by sampling N=20 diverse solutions and computing answer entropy. This is ATD.

Experiment 1: Results

EIG-uniform wins clearly

Confirms H1 — explicit reasoning about solutions outperforms implicit reasoning about questions.

EIG-logprobs underperforms

LLM's internal distribution p_ϕ is biased toward 'popular' solutions, not uniform. Using raw logprobs imports this bias into EIG estimates.

implicit-ToT helps but not enough

Seeing more candidate questions and picking among them helps. But without evaluating against solutions, selection is still guesswork.

Gap narrows for GPT-4o-mini

More capable models are better at implicit meta-cognitive reasoning. The proposed method's advantage is largest for weaker models.

What Actually Drives Performance?

Three supporting studies probing the ATD design choices

Study 1: Questions genuinely partition the solution space

Evaluated on a hand-curated list of 500 animals. EIG-uniform questions reduce compatible solutions faster than any baseline across all 6 iterations measured. The N=20 sample EIG is a good proxy for ground-truth EIG on 500 animals.

Study 2: Diversity of samples matters

Adding 'generate a diverse and representative set' to the solution-generating prompt measurably improves EIG estimates. Diverse samples better cover H, leading to more accurate partitioning estimates.

Study 3: Filtering hallucinations is crucial

As requirements accumulate, LLMs increasingly hallucinate solutions that violate prior requirements. A self-reflection filtering step — re-checking each sample against all requirements — significantly improves accuracy. 2 filter steps > 1 > 0.

Experiment 2: Active Code Generation

Real-world application with an external evaluator — near-zero noise

Setup: generate a Python function from an ambiguous description. Ask clarifying questions (as test cases) to narrow the specification. Evaluate on HumanEval and APPS benchmarks.

Questions are test cases

Instead of free-text questions, the agent generates assertion test cases: 'assert search([1,2,3], 2) == 1'. Binary (True/False) or open (expected output).

External Python interpreter answers

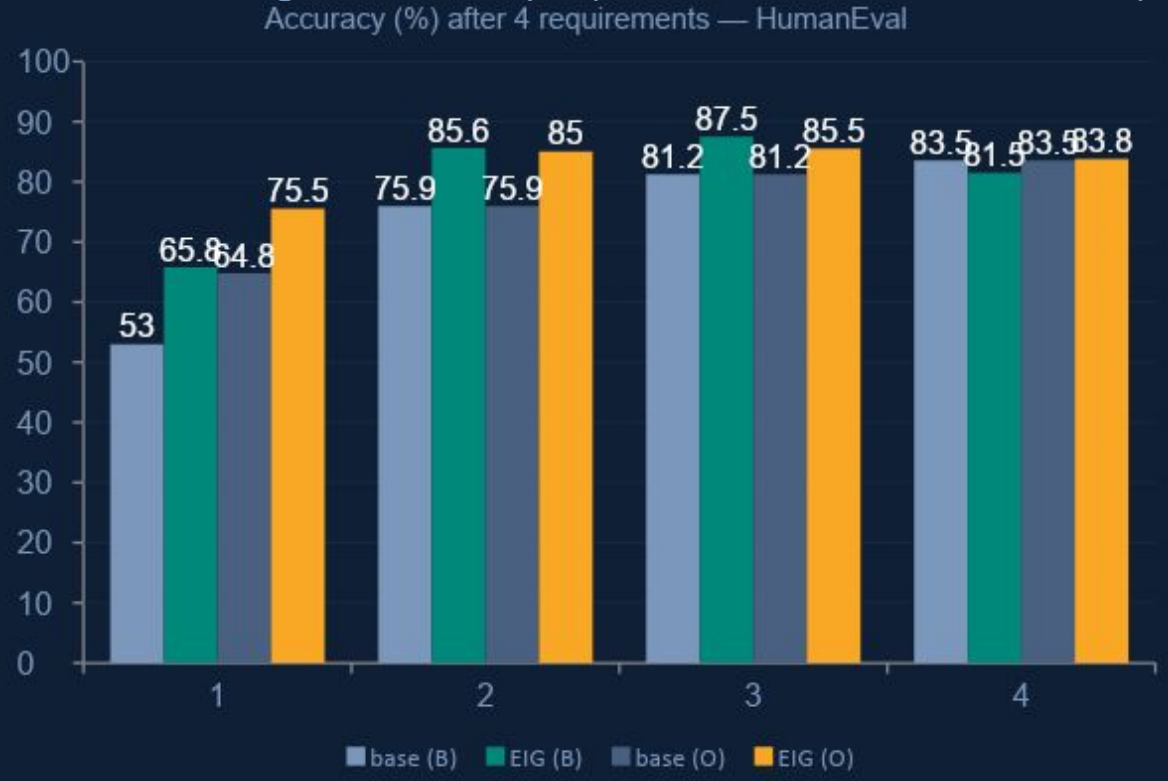
Pseudo-answers are obtained by executing candidate code solutions against the test case. This gives near-zero noise — no LLM hallucination in the evaluation step.

Two question types tested

Binary (B): assertion True/False — 2 possible answers. Open (O): input→output pair — many possible answers, higher potential information gain.

Experiment 2: Results

EIG-based strategies consistently improve code correctness — fewer questions, higher accuracy



Open > Binary consistently

More answer options = higher potential EIG. Binary questions are limited to 1 bit of info.

EIG helps all model sizes

Even GPT-4o-mini benefits. Weaker models (Llama-8B) benefit most — from 53% to 75.5%.

Consistent on APPS too

Results hold on harder competition-level problems, not just introductory HumanEval.

Only 4 questions needed

Dramatic accuracy improvement from just 4 targeted clarifying questions.

Critical View: What Are We Assuming?

Stepping back from the results to scrutinize the foundations

Uniformity assumption is strong

The EIG formula assumes p* is uniform over compatible solutions. In reality users have preferences — some solutions are much more likely than others. The method ignores these prior probabilities entirely.

Sample quality bottleneck

EIG accuracy depends on how well N=20 samples cover H. If the LLM can't generate diverse samples — especially in niche domains — the EIG estimate is unreliable regardless of the algorithm.

Oracle assumes perfect truthfulness

The theory assumes users always answer correctly. Real users answer 'I don't know', give inconsistent answers, or misunderstand questions. The paper acknowledges this as noise but doesn't model it.

No ambiguity detection

The method doesn't determine whether the task is actually ambiguous. It asks questions regardless. A task that seems ambiguous might just be hard for the LLM — asking won't help if the LLM's samples are wrong.

Critical View: Cost and Real-World Scope

The cost is O(N + M + N·M) LLM calls per iteration

N=20 solutions + M=5 questions + 20×5=100 answer evaluations = 125 LLM calls per round
Compare: baseline implicit = 1 LLM call per round. That's a 125× cost increase.

Code generation is unusually clean	Code can be executed to get answers — near-zero noise. Most real-world tasks (medical, legal, creative) have no equivalent evaluator. Extending ATD to these domains requires LLM-based evaluation, reintroducing hallucination noise.
When does it stop asking?	The paper sets a fixed number of rounds (T). There's no principled stopping criterion — when is the solution space small enough? The paper leaves this as future work.
User patience has a limit	125 LLM calls for 1 question means the latency per round is high. Users may prefer a slightly worse answer immediately over a better answer after multiple back-and-forths.
Binary vs. open question tradeoff	Open questions give more information gain but require more user effort to answer. The right balance is application-dependent and the paper doesn't provide guidance on choosing.

Where This Fits in the Bigger Picture

Active Learning	ATD is a form of active learning — selecting queries that maximize information gain. The paper generalizes classical AL to the unconstrained natural language output space of LLMs, where there's no fixed pool of labeled examples.
Tree of Thoughts	ToT and ATD both use structured sampling to improve LLM reasoning. ToT samples multiple reasoning paths; ATD samples multiple solutions. ATD's key advance: evaluating against an objective criterion (EIG) rather than asking the LLM to self-select.
RLHF / DPO	ATD is complementary to RLHF — not competing. RLHF improves population-level alignment; ATD handles individual user deviations in real time. They could be used together.
RAG & Tool-Augmented LLMs	The code generation version of ATD uses a Python interpreter as an external tool — placing ATD naturally in the emerging paradigm of tool-augmented LLMs where external evaluators validate outputs.

Main Takeaways

01

Ambiguity is the norm, not the exception

Real-world LLM usage involves systematically underspecified tasks. Treating ambiguity as an edge case is wrong — it needs to be a first-class design consideration.

02

Bootstrap weak skills with strong ones

LLMs are poor at meta-cognitive question generation but excellent at solution generation. ATD exploits this asymmetry to dramatically improve question quality.

03

EIG-uniform beats implicit reasoning

In both experiments, across all model sizes, explicit EIG computation outperforms asking the LLM to introspect. The gap is largest for weaker models.

04

External grounding eliminates noise

The code generation experiment shows that when you can ground evaluation in an external tool, ATD approaches near-optimal question selection. Domain determines what's possible.

05

Four questions can double performance

On Llama-3-8B: 35% → 75% accuracy with just 4 EIG-selected open questions. The leverage of targeted clarification is enormous.

Discussion Questions

Q1 The uniformity assumption ignores user priors. How would you extend ATD to incorporate prior knowledge about what users typically want — without requiring expensive fine-tuning?

Q2 ATD asks clarifying questions when the task is ambiguous. But how do you know when the task is ambiguous versus when it's simply hard? How would you design an ambiguity detector?

Q3 The code experiment avoids LLM noise by using a Python interpreter. What other domains have cheap, reliable external evaluators? What domains are fundamentally stuck with LLM evaluation?

Q4 The paper compares against implicit and implicit-ToT baselines. What other baselines feel missing? Could fine-tuning LLMs specifically to ask clarifying questions outperform ATD?

Q5 ATD requires multiple rounds of dialogue. In what application contexts is this natural and in what contexts does it break down? How does the user experience design change what's possible?

Ask Better Questions.

When the problem is ambiguous, the answer is not to guess — it is to ask.

And to ask strategically — selecting questions that split the solution space,
not questions that feel natural but teach you nothing.

Code: github.com/kasia-kobalczyk/active-task-disambiguation

Paper: ICLR 2025 · arXiv:2502.04485

Kobalczyk · Astorga · Liu · van der Schaar — University of Cambridge

Appendix: Why EIG-logprobs Fails

The LLM's own distribution is not uniform over H — and that's the problem

EIG-logprobs assumes:

The LLM's token log-probabilities for solutions reflect the true distribution over H.

So $p_\phi(\text{solution}) \approx p^*(\text{solution})$

Therefore: use p_ϕ as weights when computing EIG.

Seems reasonable — the LLM knows what's likely...

Why it actually fails:

p_ϕ is biased toward common solutions in training data — popular animals, standard code patterns.

Rare but valid solutions (pangolins, unusual edge cases) get near-zero probability.

This makes the effective sample non-uniform and highly biased.

EIG computed with biased weights gives incorrect partition estimates.

Result: selects questions that look good under p_ϕ's biased view, not under the true uniform view.

Appendix: The Stopping Criterion Problem

An open problem the paper leaves for future work

When have you asked enough questions?

The paper uses a fixed T rounds. In practice you need to stop when H is small enough. But how do you measure 'small enough' without knowing H* exactly?

Fixed T rounds	Simple but wasteful — may ask unnecessary questions or stop too early. Not adaptive to task difficulty.
Sample entropy threshold	Stop when the entropy of N sampled solutions falls below a threshold — i.e., when samples are converging. Principled but requires tuning the threshold.
EIG drops below threshold	Stop when the best available question has EIG < ε — diminishing returns. But ε is task-dependent.
User confirms	Ask the user 'is there anything else I should know?' This shifts burden back to user but is explicit and controllable.

Appendix: Future Direction — Synthetic Fine-tuning Data

The paper's final insight: ATD can generate training data for better implicit reasoning.

Why implicit reasoning is weak: LLMs see few examples of optimal clarifying questions in training data. ATD can generate (problem statement, optimal question) pairs at scale. Fine-tune on these → LLMs that reason implicitly better.

1. Sample diverse ambiguous problem statements
2. Run ATD to find the optimal clarifying question for each
3. Collect (problem, optimal question) pairs as supervised fine-tuning data
4. Fine-tune an LLM on this dataset
5. The fine-tuned LLM now asks better questions implicitly — without running ATD at inference

Appendix: Full Results Summary

Experiment	Model	Base acc.	EIG acc.	Gain
HumanEval (B)	Llama-3-8B	53.0%	65.8%	+12.8%
HumanEval (O)	Llama-3-8B	64.8%	75.5%	+10.7%
HumanEval (B)	GPT-3.5	70.8%	85.6%	+14.8%
HumanEval (O)	GPT-3.5	75.9%	85.0%	+9.1%
HumanEval (B)	Llama-3-70B	78.8%	87.5%	+8.7%
HumanEval (O)	GPT-4o-mini	83.5%	83.8%	+0.3%
APPS (O)	GPT-3.5	69.7%	77.3%	+7.6%
APPS (O)	GPT-4o-mini	75.1%	87.2%	+12.1%
20Q rank @10	GPT-3.5	4.9	6.2	+1.3
20Q rank @10	GPT-4o-mini	5.5	6.8	+1.3

(B) = Binary questions, (O) = Open questions. All at T=4 requirements elicited.