A Appendix

A.1 Agent Implementation

Our agent implementation is built upon the ReAct framework (Yao et al., 2023), which combines reasoning and acting in a unified architecture. We implement three distinct agent configurations to systematically evaluate different capability combinations:

Configuration 1: Answer-only: The agent directly generates responses using its internal knowledge without external information gathering. This configuration serves as a baseline to measure pure knowledge recall capabilities on ambiguous queries.

Configuration 2: Answer+Search: The agent can perform web search actions to retrieve external information before generating answers. Available actions include:

- `search(query)`: Performs web search with the specified query
- `answer(response, confidence)`: Provides final answer with confidence score

Configuration 3: Answer+Search+Ask: The full interaction-enabled agent that can additionally request clarification from users. This configuration adds:

- `ask(question)`: Poses yes/no questions to gather missing information

Action Space Design - Each agent operates with a maximum of 10 rounds, where each round allows exactly one action. The agent maintains an internal memory of previous actions and observations. For forced interaction experiments, we implement a constraint requiring minimum interaction thresholds before answer generation is permitted.

The complete system prompts and interaction protocols are detailed below.

Prompt

```
SYSTEM_PROMPT = """
## Goal
You are an intelligent agent, designed to answer user's question.
In each round, you can execute one action, and you can get the action's result as
↪  observation.
You should think step by step, and output the action you want to execute.

### Evidence first
Before answering, you MUST:
1. Identify ALL missing information dimensions (time, scope, context, conditions etc.)
2. Systematically gather evidence for each dimension
3. Verify key assumptions through multiple sources/questions
4. Only answer when you can confidently justify each part of your response

**Critical**: Most questions have hidden complexities. Your initial understanding is
↪  likely incomplete.

### Using ask
When the ask action is available, you may pose closed-ended questions to fill gaps such
↪  as time, scope, conditions, relationships, or quantities.
- Do **not** ask the user to confirm a complete candidate answer or entity name.
↪  request neutral attributes or other missing evidence instead.

**Important: When you choose the ask action, you can only ask closed-ended, yes/no
↪  questions. The user will only respond with "yes", "no", or "I don't know".**

## Available actions:
{actions}

## Output Format
When you output the action,
you should output the action name and parameters in the json format, and only one
↪  action.
Such as,
```json
{{
    "action": "",
    "params": {{
        "<param_name>": "<param_value>"
```
```

```
    }}
}}
```
Before output, you should think step by step.

Question
{question}
"""

ACT_PROMPT = """
Memory
{memory}

Observation
Last action: {last_action}
Observation: {last_observation}

Question
{question}

Action
You should output the action you want to execute.
Output your next action in JSON format, e.g.
```json
{{
    "action": "",
    "params": {{
        "<param_name>": "<param_value>"
    }}
}}
```

ROUNDS
Current round: {round_info}
You have only one opportunity to provide your final answer.
Use your remaining rounds wisely to collect evidence and test your theories before committing to an answer.
The above shows your remaining action rounds.
"""

FINAL_ROUND_ACT_PROMPT = """

Given the question and information you have gathered, output the final answer.

Round
{round_info}

Memory
{memory}

Question
{question}

Action
You should output the answer action, you can think step by step before you output the answer.
Return the final answer action in JSON, for example:
```json
{{
    "action": "answer",
    "params": {{
        "answer": "<param_value>",
        "confidence": "<param_value>"
    }}
}}
```

"""

A.2 RESPONDER SIMULATION

We implement a controlled responder simulation using GPT-4o (temperature=1.0) that provides structured feedback when agents employ the *ask* action. Upon receiving agent queries, the responder evaluates questions against available context and responds with one of three standardized options: "yes", "no", or "I don't know". The responder state s_r consists of the given context and interaction history, with transitions $T_r : (s_r, q_{agent}) \rightarrow o_r \in \{\text{yes}, \text{no}, \text{unknown}\}$ conditioned on context-

question alignment. While maintaining response diversity through LLM generation, the constrained output format ensures evaluation consistency.

The complete responder prompts are detailed below.

Prompt

```
RESPONDER_PROMPT = """
You are a specialized Q&A agent. Think step by step before you output the answer.

Rules:
- Reply with exactly one of: yes, no, or i don't know.
- Treat the context as the entire truth.
- Use only the provided CONTEXT to judge the yes/no question.
- Answer **yes** only if the context clearly states the proposition is correct.
- Answer **no** if the context contradicts the proposition (for example it states an
↪  incompatible attribute).
- If the context neither confirms nor denies it, answer **i don't know**.
- Do not rely on outside knowledge, analogies, or multi-hop guesses. Compare the
↪  relevant words directly.

CONTEXT
{context}

QUESTION
{question}

Output: yes | no | i don't know
"""
```

A.3 Evaluation Protocol

We validate simulation reliability through repeated sampling across identical context–question pairs across $k = 3$ trials, indicating stable behavior despite the stochastic generation process. We employ GPT-4o (temperature $= 0.0$) as our grader, providing ground truth, agent response, and question context for binary correctness judgments. Grader reliability is validated through spot-checking against human evaluation.

The complete responder prompts are detailed below.

Prompt

```
GRADING_PROMPT = """
\nYou are an impartial grader.

Question: {question}
Predicted Answer: {predicted_answer}
Correct Answer: {correct_answer}

CRITICAL GRADING INSTRUCTIONS:
1. The predicted answer must match the CORRECT ANSWER
2. Look for EXACT name matches or clear references to the same entity
3. Consider different languages, translations, or alternative names as potential
↪  matches
4. Be strict: partial matches or vague similarities should be 'no'

IMPORTANT: Give ONLY one score:
- 'yes': The predicted answer correctly identifies the same entity as the correct
↪  answer
- 'no': The predicted answer is wrong, matches the popular answer, or refers to a
↪  different entity

Respond with ONLY 'yes' or 'no', nothing else."""
```