AMOR: A Recipe for Building Adaptable Modular Knowledge Agents Through Process Feedback

Anonymous ACL submission

Abstract

The notable success of large language models (LLMs) has sparked an upsurge in building language agents to complete various complex tasks. We present AMOR, an agent framework based on open-source LLMs, which reasons with external knowledge bases and adapts to specific domains through human supervision to the reasoning process. AMOR builds reasoning logic over a finite state machine (FSM) that solves problems through autonomous executions and transitions over disentangled modules. This allows humans to provide direct feedback to the individual modules, and thus naturally forms process supervision. Based on this reasoning and feedback framework, we develop AMOR through two-stage fine-tuning: warm-up and adaptation. The former fine-tunes the LLM with examples automatically constructed from various public datasets and enables AMOR to generalize across different knowledge environments, while the latter tailors AMOR to specific domains using process feedback. Extensive experiments across multiple domains demonstrate the advantage of AMOR to strong baselines, thanks to its FSM-based reasoning and process feedback mechanism.

1 Introduction

Large language models (LLMs), with astounding performance over general natural language processing (NLP) problems (Wei et al., 2022a; Achiam et al., 2023; Touvron et al., 2023), have spurred great interest in building LLM-based agents to solve complex tasks by interacting with external resources such as web knowledge (Nakano et al., 2021), specialized tools (Schick et al., 2023), etc.

We focus on developing agents for knowledge-intensive tasks, where the agent completes users' information-seeking requests by interacting with specific knowledge bases (Lewis et al., 2020). To address the complexity of such tasks, we posit the desiderata for a qualifying agent as follows: Firstly, the agent should possess a robust *reasoning logic* about the task to solve individual problems with precise pathways. Secondly, the agent should maintain an *adaptive mechanism* to adjust to specific environments, rather than staying static. Thirdly, the reasoning process should be amenable to human interventions, enabling humans to steer the agent's behavior through direct *feedback* to the process rather than only to the outcome (Uesato et al., 2022). This ability can significantly facilitate alignment between agent behavior and human intent.

Although extensive studies have been conducted on building language agents, few, if any, can fulfill all the required criteria due to their uncontrollable reasoning logic, static model capability, or sparse/missing feedback signals, as detailed in Tab. 1. Consequently, it is still challenging for users to critique, and thus guide existing agents to follow targeted manners, especially when the agents are built upon less powerful LLMs (Liu et al., 2023b).

We introduce an **A**daptable **MO**dula**R** knowledge agent (**AMOR**) that can reason and adapt, with the reasoning process amenable to human supervision, based on open-source LLMs. AMOR's reasoning logic is formalized as a finite state machine (FSM) (Clarke et al., 1986; Lee and Yannakakis, 1996) that solves problems via a series of executions and transitions over a set of modules (Fig. 1). This naturally enables the desired process-based supervision mechanism, allowing users to give feedback to each LLM-controlled module. AMOR supports flexible forms of feedback, either binary judgments regarding the correctness or refinement of the outputs. The reasoning logic and process feedback mechanism together frame how AMOR thinks, acts, and interacts with users and task environments.

The training in AMOR happens in two stages: **(1) Warm-up:** the modular design enables us to construct training data separately for each disentangled module without requiring complete trajectories for specific tasks. As a result, we create a

Method	Reasoning Logic		Adaptive Mechanism	Feedback
	Step	Inter-Step Dependency		
WebGPT (Nakano et al., 2021)	Tool Invoking	*Undefined*	Imitation Learning from Humans	Outcome
CoT (Wei et al., 2022b)	Reasoning	*Undefined*	Prompting	*Undefined*
ToT (Yao et al., 2023a)	Reasoning	*Undefined*	Prompting	Process
ReAct (Yao et al., 2023b)	Reasoning&Tool Invoking	*Undefined*	Prompting	*Undefined*
Reflexion (Shinn et al., 2023)	Reasoning&Tool Invoking	*Undefined*	Prompting	Process
AgentLM (Zeng et al., 2023)	Reasoning&Tool Invoking	*Undefined*	Imitation Learning from LLMs	Outcome
MetaGPT (Hong et al., 2023)	Specialized Module	Pipeline	Prompting	Process
LUMOS (Yin et al., 2023)	Specialized Module	Pipeline	Imitation Learning from Humans	*Undefined*
AMOR	Specialized Module	Finite State Machine	Exploration&Exploitation	Process

Tab. 1: Comparison between AMOR and representative methods for building agents.

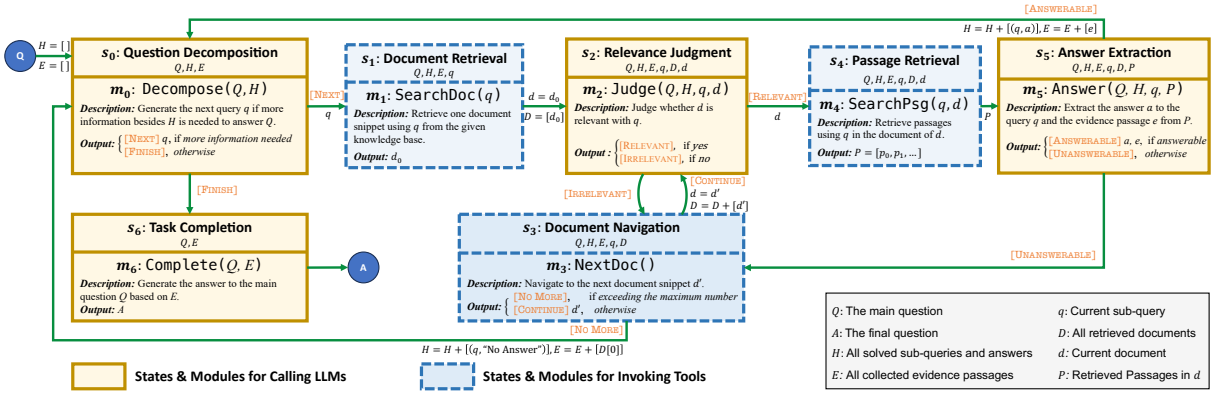

Fig. 1: AMOR's state transition diagram. Each box represents a state and the corresponding module that is executed when entering the state. There may be multiple categories of execution results distinguished by special branch tokens such as "[NEXT]." Then AMOR determines the next state based on the branch tokens.

large dataset of 50k examples covering multiple distinct tasks, simply using public datasets. We fine-tune AMOR on this data for generalization over various knowledge-seeking scenarios. **(2) Adaptation:** when deployed, we tailor AMOR to the target domain by letting it autonomously address user tasks (i.e., exploration), collecting process feedback for each LLM output, and selecting those outputs that users judge as right or refine to be right for further fine-tuning (i.e., exploitation).

Our contributions are summarized as follows:

I. We propose a general framework for building knowledge agents, featuring FSM-based reasoning logic and a process feedback mechanism. We focus on text corpora as knowledge bases, but the approach can be flexibly extended to other knowledge types and user tasks by customizing the modules and dependencies within the FSM framework.

II. Experiments across multiple domains show the strong advantage of the FSM-based reasoning logic with 30%-40% improvements over baselines when based on off-the-shelf LLMs (e.g.,GPT-4[1]).

III. Switching to fine-tuned LLMs, the warm-up stage empowers AMOR to generalize to multiple domains and surpass strong baselines. After we adapt AMOR to specific domains, Subsequent domain-specific adaptations reveal that process feedback is significantly more effective in improving the reasoning process than outcome feedback[2].

2 Related Work

Language Agents. Interest is surging in building agents for tasks necessitating multi-step reasoning. Existing work falls into two groups. The first group focuses on designing agent architectures, such as CoT's step-by-step reasoning (Wei et al., 2022c) and ReAct's integration of reasoning, action, and observation to allow tool use (Yao et al., 2023b). Nevertheless, such free-form reasoning constraints human intervention. In contrast, modular agents follow a pipeline to execute specialized modules (Khot et al., 2023; Hong et al., 2023; Gur et al., 2023; Besta et al., 2023; Yin et al., 2023), improving the ease of intervention. The second group aims to design adaptive mechanisms

[1]In this work, GPT-3.5/GPT-4 refers to the OpenAI's API "gpt-3.5-turbo" and "gpt-4-1106-preview," respectively.

[2]The code and data will be publicly available.

for adapting agents to specific scenarios. ToT (Yao et al., 2023a) and Reflexion (Shinn et al., 2023) use environment feedback for multi-path pruning and iterative single-path refinement, respectively, but suffer from poor inference efficiency and need for real-time feedback. As a fine-tuning approach, recent work equipped open-source LLMs with specific agent abilities by learning trajectories from humans (Nakano et al., 2021) or GPT-4 (Zeng et al., 2023; Chen et al., 2023), with correctness validation through outcome feedback. In contrast, our modular agent AMOR employs FSM-based reasoning with a stronger capacity for handling complex tasks than simple pipelines and adapts effectively to specific environments via process feedback.

Retrieval-Augmented Generation (RAG). The RAG paradigm augments the inputs of LLMs with retrieved passages to enhance factuality (Guu et al., 2020; Lewis et al., 2020). Recent studies have developed interleaved reasoning-retrieval for better information recall than one-step retrieval (Trivedi et al., 2023; Jiang et al., 2023; Press et al., 2023). However, retrieval may introduce noise that leads to low-quality answers (Shi et al., 2023). To tackle this, Self-RAG (Asai et al., 2023) trained LLMs to selectively perform retrieval and utilize retrieved passages. Unlike RAG approaches, AMOR emphasizes an explainable reasoning process for proactively decomposing questions and seeking evidence for grounded generation, and allows for process feedback from humans. However, RAG mainly focuses on integrating parametric factual knowledge in LLMs and retrieved non-parametric knowledge, which is less explainable and intervenable.

3 AMOR Agent

AMOR relies on three key techniques: FSM-based reasoning logic, a process feedback mechanism, and a two-stage fine-tuning strategy. We detail the definition of the reasoning logic and its specification assuming the knowledge base is a text corpus in §3.1, the method for fine-tuning open-source LLMs as a warm-up stage in §3.2, and the adaptation stage driven by process feedback in §3.3.

3.1 Reasoning Logic

FSM-based reasoning logic can be generally defined by a quadruple: $\{\mathcal{S}, \mathcal{M}, \mathcal{E}, \mu\}$, where

- $\mathcal{S} = \{s_0, \ldots, s_{N-1}\}$ is a set of states with s_0 as the initial state and s_{N-1} as the final state. Each state holds variables to track context information.

Algorithm 1 FSM-based Reasoning Logic

Input: Agent at the state $s = s_0$; Q: Question.
Output: A: Final Answer; R: Reasoning Process.

$R = [\,]$
while True **do**
  $y = m(s)$ // Obtain the output y given s from the corresponding module m.
  R.append({"state": s, "output": y})
  if $s = s_{N-1}$ **then**
    $A = y$ // Reach the final state.
    break
  $s = \mu(s, y)$ // Transit to the next state.
return A, R

- $\mathcal{M} = \{m_0, \ldots, m_{N-1}\}$ is a set of modules with m_k triggered when the reasoning flow reaches state s_k. $\mathcal{M}$ includes (a) Tool modules ($\mathcal{M}_{\text{TOOL}}$) for invoking tools, and (b) LLM modules ($\mathcal{M}_{\text{LLM}}$) for calling LLMs. When customizing FSM for a specific task, one can design $\mathcal{M}$ by following two principles: **(1) Single Responsibility Principle.** Each module handles one specific sub-task, which can be determined by manually decomposing the main task until the tool or LLM performs well for each sub-task or it is difficult even for human experts to further decompose. **(2) Least Dependency Principle.** Each module depends on as few historical steps as possible to avoid distraction induced by unnecessary information.
- $\mathcal{E}$ is the set of all possible outputs of $\mathcal{M}$.
- $\mu : \mathcal{S} \times \mathcal{E} \rightarrow \mathcal{S}$ is the transition function that determines the next state of the reasoning flow given the current state and the execution result of the corresponding module.

Alg. 1 outlines the application of FSM-based reasoning logic for deducing the answer A with the reasoning process R for an input question Q.

When the external knowledge base is a text corpus, an instantiation of the reasoning logic can be represented by the state transition diagram in Fig. 1. In this case, $\mathcal{M}_{\text{TOOL}}$ perform document and passage retrieval using external retrievers; while $\mathcal{M}_{\text{LLM}}$ leverage the LLM to analyze and digest the question, documents, and passages to deduce the final answer. To distinguish different types of outputs from a module that requires different subsequent modules, we employ a set of special branch tokens such as "[NEXT]" to guide μ to determine the next state. In summary, AMOR answers question Q by **(1)** iteratively decomposing Q to a sub-query q at state s_0, and finding the answer a to q and the evidence passage e through iterative knowledge retrieval, relevance evaluation, retrieval refinement (i.e., "Passage Retrieval"), and answer extrac-

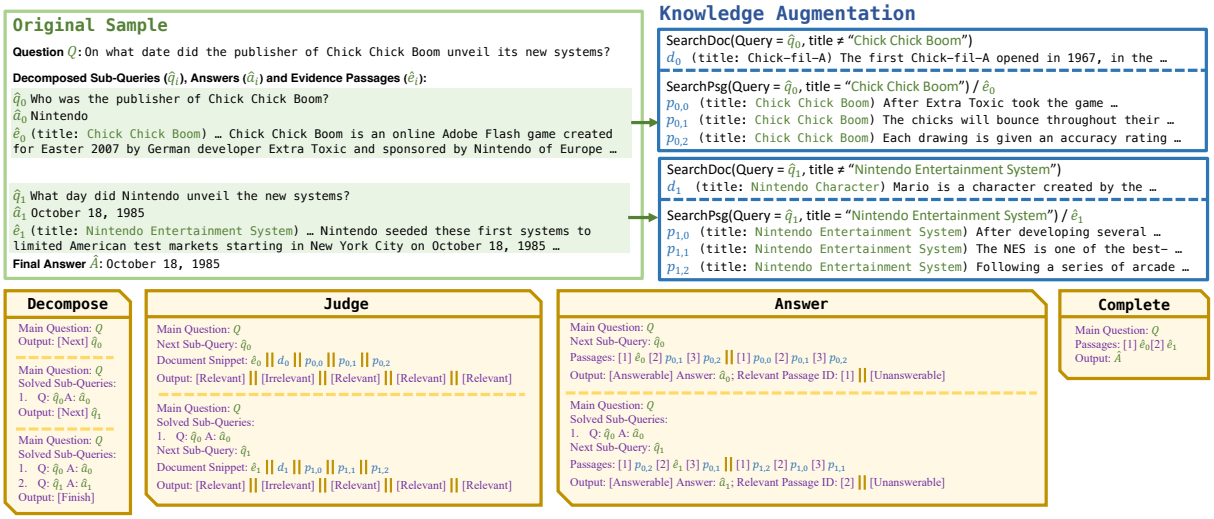

Fig. 2: On the top left is a sample question from Musique (Trivedi et al., 2022), providing ample information (in **green**) for constructing training examples for four LLM modules of AMOR (bottom). We augment extra knowledge (in **blue**) for the Judge and Answer module by invoking the SearchDoc and SearchPsg tools (top right). In each example, we highlight the prompt in **purple** to format the current state (before "Output:") and output (after "Output:"), and use "||" to separate different examples for training.

tion, until no more knowledge is needed; and **(2)** deducing the final answer A based on the collected evidence passages at the final state. Appendix A.1 details the full algorithm and prompts of AMOR.

Defining reasoning logic as an FSM offers three advantages: **(1) Structured Thinking.** FSM makes specifications of inter-step dependencies (e.g., prioritization, branch selection) easy, and thus enables narrowing down the exploration space. **(2) Skill Disentanglement.** By decomposing complex tasks into modular steps, one can independently construct training data for each module, which significantly reduces the difficulty of implementing AMOR with open-source LLMs (cf., §3.2). This feature also allows AMOR to focus on single steps, thereby mitigating the weakness of LLMs in reasoning over long context formed by task-solving trajectories (Liu et al., 2023a). **(3) Intervenable Workflow.** The structured reasoning process enables users to easily diagnose the agent's mistakes and provide process feedback for improving the reasoning capability of the agent (§3.3).

3.2 Warming Up Open-Source LLMs

Open-source LLMs are observed to fall short in complex agent tasks (Xu et al., 2023; Liu et al., 2023b). Recent studies have improved their reasoning abilities through imitation learning using trajectories from advanced LLMs such as GPT-4 (Zeng et al., 2023; Chen et al., 2023). However, even GPT-4 can struggle with producing high-quality reasoning trajectories (Qin et al., 2023).

AMOR's modular design enables us to construct training data for each module separately from existing datasets without simulating the whole trajectories, thus greatly alleviating the above issue. Formally, given a sample question Q with annotations of the final answer $\hat{A}$, all sub-queries and answers $\hat{H} = [(\hat{q}_0, \hat{a}_0), (\hat{q}_1, \hat{a}_1), \cdots]$, and all evidence passages $\hat{E} = [\hat{e}_0, \hat{e}_1, \cdots]$, we can directly transform these annotations into a suitable format to serve as training data for Decompose and Complete in Fig. 1. Since Judge and Answer require multiple types of retrieved knowledge (e.g., *relevant* or not), we employ retrieval tools to augment the input. Fig. 2 exemplifies the construction pipeline, which can be easily extended to other knowledge-intensive datasets and specific domains. Appendix A.3 shows more details.

Then, we fine-tune open-source LLMs with the standard language modeling loss[3]:

$$\mathcal{L}_1 = \sum_{\substack{m \in \mathcal{M}_{\text{LLM}}, \\ (\hat{s},\hat{y}) \in \mathcal{D}_m}} -\lambda_m \log P_\theta(\hat{y}|\hat{s}), \qquad (1)$$

where θ denotes the parameters, $\mathcal{D}_m$ is the collection of training examples for module $m \in \mathcal{M}_{\text{LLM}}$, $(\hat{s}, \hat{y})$ is a state-output pair from $\mathcal{D}_m$, and $\{\lambda_m\}$ are hyper-parameters to balance different modules.

[3]We fine-tune one LLM in a multi-task fashion. Another option is fine-tuning different LLMs for different modules like Yin et al. (2023) with higher deployment cost.

3.3 Adaptation through Process Feedback

Feedback is crucial for adapting language agents to specific environments (Wang et al., 2023a), especially when dealing with novel, rare, or evolving domain knowledge. Prior agents commonly used outcome feedback for adaptation which assesses the correctness of intermediate steps based on the success or failure of the outcome (Zeng et al., 2023; Chen et al., 2023). However, outcome feedback is too sparse to improve intermediate reasoning (Lightman et al., 2023). Recent studies also highlighted that LLMs' reasoning steps are likely to contradict the outcome (Liu et al., 2023c), which means that outcome feedback may inevitably introduce noise during training (see examples in Appendix B.7). In contrast, AMOR's process feedback mechanism can effectively alleviate these issues.

Alg. 2 describes the adaptation mechanism of AMOR parameterized by θ, specifically as three steps: **(1) Exploration.** AMOR$_\theta$ answers the input question Q by interacting with a knowledge base. **(2) Feedback Collection.** AMOR's reasoning process for Q is evaluated with feedback f for each LLM output y, which is either "right/wrong" or a refined version of y. We discard outputs labeled as "wrong" and determine the feedback-refined target output $\tilde{y}$ for the remaining outputs as follows:

$$\tilde{y} = \begin{cases} y & \text{if } f = \text{ "right"}, \\ f & \text{if } f \text{ is the refinement of } y. \end{cases} \quad (2)$$

(3) Exploitation. Every t iterations of the former two steps, we fine-tune AMOR$_\theta$ with the loss:

$$\mathcal{L}_2 = \sum_{\substack{m \in \mathcal{M}_{\text{LLM}}, \\ (s,\tilde{y}) \in \mathcal{R}_m}} -\lambda_m \log P_\theta(\tilde{y}|s), \quad (3)$$

where $\mathcal{R}_m \subseteq \mathcal{R}$ denotes the training examples for module m. The adaptation mechanism is also compatible with other algorithms for exploiting process feedback, such as PPO (Schulman et al., 2017), unlikelihood-training (Welleck et al., 2019), DPO (Rafailov et al., 2023), etc., which however is beyond the scope of this paper.

Algorithm 2 Adaptation through Process Feedback

Input: AMOR$_\theta$: Initial Policy; T: Exploration Steps between Exploitation; I: Number of Iterations.
Output: AMOR$_\theta$: Adapted Policy.

while $i \leftarrow 1$ **to** I **do**
  $\mathcal{R} = [\,]$ // Feedback-Refined Reasoning Processes
  while $t \leftarrow 1$ **to** T **do**
    // Exploration
    Receive an input question Q.
    Collect AMOR$_\theta$'s reasoning process R. // Alg. 1
    // Feedback Collection for Each LLM Module
    foreach $r \in R$ **do**
      Extract the state s and output y from r.
      if The corresponding module $m \in \mathcal{M}_{\text{LLM}}$ **then**
        Collect feedback f for s and y.
        if $f \neq$ "wrong" **then**
          Refine y to $\tilde{y}$ based on f. // Eq. 2
          $\mathcal{R}$.append({"state": s, "output": $\tilde{y}$})
  // Exploitation
  Optimize θ to minimize $\mathcal{L}_2$ on $\mathcal{R}$. // Eq. 3
return AMOR$_\theta$

Module	Branch Token	2Wiki	Musique	NQ	BoolQ
Decompose	[NEXT]	3,500	3,500	500	500
	[FINISH]	500	500	500	500
Judge	[RELEVANT]	2,000	2,000	2,000	2,000
	[IRRELEVANT]	2,000	2,000	2,000	2,000
Answer	[ANSWERABLE]	500	3,000	1,500	3,000
	[UNANSWERABLE]	500	1,000	1,000	1,000
Complete	-	3,000	4,000	1,500	4,000
Overall	-	*12,000*	*16,000*	*9,000*	*13,000*

Tab. 2: Statistics of the warm-up data.

4 Experiments

4.1 Setup

Tools Modules. We construct retrievers for both SearchDoc and SearchPsg using Contriever-MS MARCO (Izacard et al., 2022). SearchDoc retrieves a single document snippet per query, while SearchPsg fetches the top three relevant passages from a given document. By invoking NextDoc, at most nine more document snippets are returned. Appendix B.1 presents more details.

Warm-Up Datasets. We employ four question-answering (QA) datasets to warm up open-source LLMs, including 2WikiMultiHopQA (2Wiki) (Ho et al., 2020), Musique (Trivedi et al., 2022), NaturalQuestions (NQ) (Kwiatkowski et al., 2019) and BoolQ (Clark et al., 2019). They vary in levels of question complexity (single- or multi-hop), answer types (phrase spans or yes/no), and types of dependency structures between sub-queries (e.g., serial or parallel), etc. Tab. 2 shows the detailed statistics.

Adaptation & Evaluation Datasets. We consider three benchmarks, by which we simulate different deployment scenarios: **(1) HotpotQA** (Yang et al., 2018): a challenging multi-hop QA dataset built on Wikipedia articles. We use the Wikipedia dump in the Contriever paper (Izacard et al., 2022) as the knowledge base. **(2) PubMedQA** (Jin et al., 2019): a biomedical QA dataset that requires answering a question by "yes/no" given a PubMed abstract. We adapt the data to retrieval-based QA

Module m	Output y	Silver Process Feedback f
Decompose(Q, H)	[NEXT] q [FINISH]	"right", if the retrieved documents using q overlap the documents corresponding to $\hat{E}$; "wrong", otherwise. "right", if $\hat{E} \subseteq E$ (i.e, evidence passages collected by AMOR); "wrong", otherwise.
Judge(Q, H, q, d)	[RELEVANT] [IRRELEVANT]	"[RELEVANT]", if one of passages in $\hat{E}$ comes from the same document as d; "[IRRELEVANT]", otherwise.
Answer(Q, H, q, P)	[ANSWERABLE] a e [UNANSWERABLE]	"right", if $e \in \hat{E}$; "wrong", otherwise "right", if $P \cap \hat{E} = \emptyset$; "wrong", otherwise
Complete(Q, E)	A	$\hat{A}$, if $\hat{E} \subseteq E$; "wrong", otherwise.

Tab. 3: Automatic annotation strategy for silver process feedback for different LLM modules. The outputs along with feedback highlighted in green will be used for exploitation of AMOR, while those in red will be discarded.

Dataset	Knowledge Base	Avg. Len	# Train	# Val	# Test
HotpotQA	Wikipedia Articles	138	2,000	100	500
PubMedQA	PubMed Abstracts	303	401	44	445
QASPER	One NLP Paper	102	700	45	382

Tab. 4: Datasets for adaptation and evaluation. **Avg. Len** refers to the average length of passages in the corresponding knowledge base, counted by the GPT tokenizer (Radford et al., 2019). **Val** is the validation set.

by piling all 274k abstracts provided in the paper as a knowledge base, where each document comprises one abstract passage. **(3) QASPER** (Dasigi et al., 2021): answering questions in free form based on a long NLP paper. For each question, we regard the corresponding paper as a knowledge base and each section of the paper as a document with several passages. Tab. 4 shows the statistics of the three datasets. We use the training and validation sets for adaptation fine-tuning and the test sets for evaluation. For evaluation metrics, we use exact match (EM) and F1 scores for HotpotQA and QASPER; and the accuracy (ACC) of "yes/no" for PubMedQA. More details are in Appendix B.2.

Feedback Annotation. We simulate human behavior and provide silver feedback to AMOR's reasoning processes based on the gold answer $\hat{A}$ and gold evidence passages $\hat{E} = [\hat{e}_0, \hat{e}_1, \cdots]$ for each target question Q, which are already included in the training and validation data of the three benchmarks. Tab. 3 shows how we annotate the feedback for each LLM output y. Note that AMOR is applicable for gold feedback from humans in realistic applications. Appendix B.3 discusses the accuracy of the silver feedback through human evaluation.

Implementation Details. We set λ_m in Eq. 1 and Eq. 3 to 1 for all modules, set $I = 1$ in Alg. 2 and T as the size of the training set for each dataset, and fine-tune LLAMA-2-7B/13B-Chat for two epochs with a learning rate of $2e^{-5}$ using 4 NVIDIA 80GB A100 GPUs. While applying AMOR for inference, we use greedy decoding for all generations. Besides, we set the maximum number of decomposed sub-queries to the maximum count of gold evidence passages, i.e., $2/1/1$ for HopotQA/PubMedQA/QASPER, respectively. Once the maximum number is reached, AMOR is transited to state s_6 ("Task Completion") to finalize the answer.

Baselines. We compare AMOR to various baselines with or without fine-tuning: **(1) CoT** (Wei et al., 2022c): it prompts an off-the-shelf LLM to generate the answer through step-by-step reasoning. **(2) One-Step Retrieval (OneR)**: it uses the question as a query to retrieve top-K document snippets with the SearchDoc module to augment the input. We set K as the maximum number of gold evidence passages in each dataset. Under the fine-tuning setting, we use the gold evidence passages for training. OneR can be viewed as an RAG implementation for a simplification of AMOR. **(3) ReAct** (Yao et al., 2023b): it interleaves thought, action, and observation steps. An action can be either invoking the retrieval tools or finalizing an answer. We also compare AMOR with fine-tuned ReAct-style agents including AgentLM (Zeng et al., 2023) and FIREACT (Chen et al., 2023). We set the maximum number of action steps to 20. **(4) Modular Agents:** ReWoo (Xu et al., 2023) follows a pipeline that plans all sub-goals, generates actions, and then executes, while LUMOS (Yin et al., 2023) applies this pipeline iteratively, tackling one sub-goal at a time with each interaction. Both agents utilize GPT-3.5 as a supplementary QA tool during action generation. Similar to AMOR, they modularize language agents; however, they lack explicit mechanisms for assessing the relevance of retrieved information. Under the setting without fine-tuning, we provide in-context examples for the baselines following their official implementations.

Furthermore, we also conduct ablation studies to investigate the influence of different components,

Module m	Target Output $\tilde{y}$ Refined Based on f_o
Decompose(Q, H)	y if $f_o = \hat{A}$; *discarded*, otherwise.
Judge(Q, H, q, d)	y if $f_o = \hat{A}$; $\neg y$, otherwise.
Answer(Q, H, q, P)	y if $f_o = \hat{A}$; *discarded*, otherwise.
Complete(Q, E)	$\hat{A}$ if $\hat{E} \subseteq E$; *discarded*, otherwise.

Tab. 5: Refining each output y to $\tilde{y}$ based on the outcome feedback f_o to adapt AMOR, where $\neg y$ denotes converting the binary output y to its opposite label. Outputs labeled as *discarded* are excluded from fine-tuning.

resulting in two more baselines: **(1) AMOR$_{\text{WFT}}$**: AMOR with only warm-up fine-tuning, without further adaptation; and **(2) AMOR$_{\text{Outcome}}$**: outcome feedback instead of process feedback is utilized in adaptation after AMOR is warmed-up. We annotate the silver outcome feedback f_o for the Complete module at the final state as $\hat{A}$ if all gold evidence passages are successfully collected (i.e., $\hat{E} \subseteq E$), and "wrong" otherwise. Then we determine the target output for all LLM modules for adapting AMOR using Eq. 3, as detailed in Tab. 5. For clarity, we denote our final method as AMOR$_{\text{Process}}$.

4.2 Main Results

Tab. 6 reports the evaluation results of AMOR and baselines on three datasets, revealing three key findings: **(1) The FSM paradigm is clearly advantageous to prior agent frameworks.** AMOR$_{\text{w/o FT}}$ delivers strong performance by improving 41.9%, 32.1%, and 41.2% over ReAct on average when built on top of off-the-shelf LLMs, including L-7B, GPT-3.5, and GPT-4, respectively. This indicates that our proposed FSM paradigm is more effective in leveraging LLMs for complex reasoning. **(2) Warm-up fine-tuning generally enhances AMOR in downstream tasks.** When based on L-7B, AMOR$_{\text{WFT}}$ outperforms AMOR$_{\text{w/o FT}}$ across all datasets. Furthermore, AMOR$_{\text{WFT}}$ also surpasses other fine-tuned ReAct-style and modular agents, even including FIREACT that is fine-tuned with in-domain HotpotQA trajectories from GPT-4. This suggests the potential of utilizing existing datasets for developing powerful agents with well-defined reasoning logic. **(3) Process feedback is more effective than outcome feedback in facilitating the adaptation of agents.** The order that AMOR$_{\text{Process}}$ > AMOR$_{\text{Outcome}}$ > AMOR$_{\text{WFT}}$ indicates the impact of feedback in terms of tailoring agent behavior to specific domains, and process feedback is more helpful than outcome feedback for leading to the correct final answers.

Method	Base LLM	HotpotQA EM	HotpotQA F1	PubMedQA ACC	QASPER EM	QASPER F1
Without Fine-Tuning						
ReAct	L-7B	12.2	16.6	61.8	6.0	19.2
AMOR$_{\text{w/o FT}}$	L-7B	26.0	34.6	62.9	4.5	21.3
CoT	GPT-3.5	28.0‡	-	*N/A*	*N/A*	*N/A*
OneR	GPT-3.5	33.4	42.1	72.6	6.8	23.3
ReAct	GPT-3.5	30.8	38.8	58.2	5.8	27.0
ReWoo	GPT-3.5	30.4†	40.1†	-	-	-
AMOR$_{\text{w/o FT}}$	GPT-3.5	39.6	49.3	68.8	10.0	30.8
CoT	GPT-4	45.0‡	-	*N/A*	*N/A*	*N/A*
ReAct	GPT-4	42.0‡	-	62.1	7.1	26.2
AMOR$_{\text{w/o FT}}$	GPT-4	**55.2**	**65.2**	**80.0**	**11.5**	**37.4**
With Fine-Tuning						
OneR	L-7B	34.8	43.8	75.3	11.0	25.5
AgentLM	L-7B	22.3†	-	64.9	4.2	20.2
FIREACT	L-7B	26.2†	-	66.1	6.5	18.4
LUMOS	L-7B	29.4†	-	70.3	7.1	19.5
AMOR$_{\text{Process}}$	L-7B	41.4	50.9	78.2	**17.8**	33.2
AMOR$_{\text{WFT}}$	L-7B	30.4	39.3	71.2	10.7	22.6
AMOR$_{\text{Outcome}}$	L-7B	37.0	45.6	75.5	9.2	24.4
AgentLM	L-13B	29.6†	-	67.9	7.1	24.4
AMOR$_{\text{Process}}$	L-13B	**44.4**	**52.5**	**79.6**	17.3	**35.8**
AMOR$_{\text{WFT}}$	L-13B	34.0	41.6	72.6	14.1	25.3
AMOR$_{\text{Outcome}}$	L-13B	39.0	48.8	78.2	10.0	26.3

Tab. 6: Results of AMOR and baselines under different settings. "L-7/13B" is short for "LLAMA-2-7/13B-Chat." We highlight the best results in **bold** and underline the second best. Results marked with † are reported in the original paper, and results marked with ‡ are reported in Chen et al. (2023). *N/A* means the method does not apply to the datasets.

4.3 Discussions

The main results have substantiated the benefits of different components of AMOR for successful task completion. Nonetheless, we are still curious about three key research questions: **(1) RQ1:** How do the AMOR variants differ in the ability to collect evidence? **(2) RQ2:** To what extent does feedback-driven adaptation enhance the AMOR's reasoning process? **(3) RQ3:** Is process feedback more data-efficient than outcome feedback for adaptation? Appendix B.5 and B.6 further demonstrate the efficient token usage of AMOR and the flexibility of AMOR's reasoning framework, respectively.

RQ1: Evidence Collection Comparison. We use recall of gold evidence passages ($\hat{E}$) among those collected by AMOR (E) to assess AMOR's ability to collect evidence, formally as $\frac{\#\{\hat{E} \cap E\}}{\#\{\hat{E}\}}$.

As shown in Tab. 7, we observe: **(1)** Warm-up fine-tuning consistently enhances evidence collection, with AMOR$_{\text{WFT}}$ achieving higher recall than AMOR$_{\text{w/o FT}}$ across all datasets. **(2)** Adaptation through outcome feedback (AMOR$_{\text{Outcome}}$) exerts a negligible impact on the recall results compared with AMOR$_{\text{WFT}}$, suggesting the superiority of AMOR$_{\text{Outcome}}$ to AMOR$_{\text{WFT}}$ in final answers (see

Method	Base LLM	HotpotQA	PubMedQA	QASPER
OneR	**N/A**	31.1	67.6	24.9
AMOR$_{\text{w/o FT}}$	**L-7B**	24.1	54.2	24.3
AMOR$_{\text{Process}}$	**L-7B**	51.3	78.7	39.5
AMOR$_{\text{WFT}}$	**L-7B**	44.3	68.1	27.0
AMOR$_{\text{Outcome}}$	**L-7B**	44.1	67.9	26.4
AMOR$_{\text{Process}}$	**L-13B**	**52.0**	**80.0**	**42.4**
AMOR$_{\text{WFT}}$	**L-13B**	44.2	69.9	27.7
AMOR$_{\text{Outcome}}$	**L-13B**	46.7	67.9	27.7

Tab. 7: Recall scores of AMOR under different settings.

Method	Decompose	Judge	Answer	Complete
AMOR$_{\text{Process}}$	**72.0**	95.5	**80.0**	**52.0**
AMOR$_{\text{WFT}}$	60.7	**96.8**	74.2	36.0
AMOR$_{\text{Outcome}}$	62.7	95.3	73.6	46.0

Tab. 8: Accuracy of four LLM modules based on the human study. All AMOR variants are based on L-7B.

Tab. 6) may stem from the improvement of Complete. **(3)** Process feedback is crucial to improve the evidence collection ability, with AMOR$_{\text{Process}}$ substantially outperforming the other variants.

RQ2: Reasoning Process Assessment. To measure the accuracy of AMOR's reasoning process, we performed a human study on the HotpotQA test set, which involved: (1) selecting 50 random questions; (2) manually annotating the gold feedback f_{human} for each LLM module output; and (3) calculating the accuracy of each LLM module output based on f_{human} (1/0 indicating "right/wrong"). More details are presented in Appendix B.3.

Table 8 presents the accuracy for three AMOR variants, affirming RQ1's findings: process feedback significantly improves the reasoning process over AMOR$_{\text{WFT}}$ that lacks adaptation, while outcome feedback has a negligible effect. Moreover, AMOR$_{\text{Process}}$ still relatively lags in the Decompose and Complete modules, hinting that future enhancements could focus on including more corresponding data during two fine-tuning stages.

RQ3: Data Efficiency for Adaptation. We aim to compare the data efficiency of different feedback types for adaptation in terms of the number of exploratory instances required. To this end, we adjust the exploration steps T in Alg. 2, selecting values at intervals of 200, ranging up to 2,000 steps on HotpotQA. Appendix B.4 further discusses the cases with $I > 1$ in Alg. 2 where AMOR is optimized over multiple rounds.

Fig. 3 shows the post-adaptation performance of AMOR varying with the number of exploratory

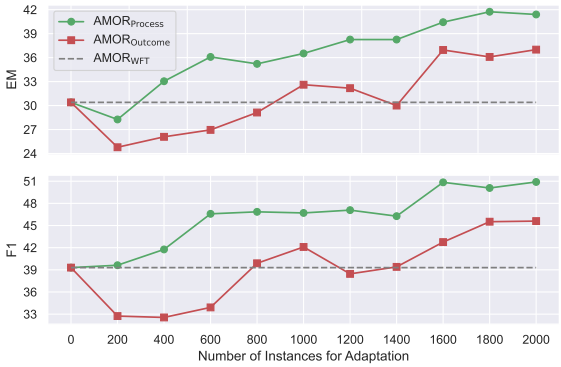

Fig. 3: EM and F1 scores on HotpotQA varying with the number of exploratory instances for adaptation.

instances (i.e., T). Compared to AMOR$_{\text{Outcome}}$, AMOR$_{\text{Process}}$ requires significantly fewer exploration steps to achieve comparable performance. Notably, AMOR$_{\text{Outcome}}$ shows a marked decline in performance when exposed to a limited number of exploratory instances (< 800), suggesting a reduced adaptability in exploration-limited scenarios. Conversely, AMOR's robust performance under such constraints highlights its superior adaptability and efficiency with minimal interaction.

4.4 Case Study

Appendix B.7 presents several examples to further illustrate AMOR's strengths in reasoning logic and intervenability, as well as the limitations of relying on outcome feedback for adaptation, emphasizing the crucial role of process feedback.

5 Conclusion

In this work, we develop AMOR, an adaptable modular agent designed for knowledge-intensive tasks, featuring FSM-based reasoning logic and a process feedback mechanism. Based on open-source LLMs, AMOR undergoes a two-stage fine-tuning: initial warm-up to generalize across task environments and subsequent domain-specific adaptation through process feedback. Extensive experiments demonstrate AMOR's advantages over strong baselines across multiple domains. Further discussions highlight the effectiveness and efficiency of process feedback in adaptation. compared to previous agents. Future work will explore extending our paradigm to more knowledge types (e.g., structured knowledge bases) and broader agent tasks, ultimately empowering LLMs to autonomously design FSM-based reasoning logic on top of our paradigm.

6 Limitations

This study has demonstrated the benefits of two components: (1) explicitly defined FSM-based reasoning logic, and (2) the process feedback mechanism. Nonetheless, a notable limitation must be acknowledged when extending our approach to other tasks. While we have made initial efforts to outline the general principles for crafting the FSM in §3.1 and show the flexibility of adapting AMOR's FSM in Appendix B.6, it still requires a human-driven design process. Looking ahead, our future work aims to enable LLMs to autonomously instantiate FSM-based reasoning logic in Alg. 1 for diverse user tasks, thereby reducing reliance on human design. Furthermore, we believe that the FSM-based reasoning logic makes it easier for humans to supervise LLMs that potentially outperform humans on the task.

Another limitation pertains to the adaptation experiments. Despite our emphasis on the significance of process feedback from real users for adapting agents to specific deployment environments, we had to rely on automatically annotated silver feedback due to practical constraints, including the scarcity of high-quality annotators and budget restrictions. To alleviate concerns regarding the use of such silver feedback, Appendix B.3 presents a thorough study regarding the adequacy of this silver feedback. We believe that our approach offers a solid foundation for the continuous evolution of post-deployment language agents.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michał Podstawski, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2023. Graph of Thoughts: Solving Elaborate Problems with Large Language Models.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. 2023. Fireact: Toward language agent fine-tuning. *arXiv preprint arXiv:2310.05915*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936.

Edmund M Clarke, Orna Grumberg, and Michael C Browne. 1986. Reasoning about networks with many identical finite-state processes. In *Proceedings of the fifth annual ACM symposium on Principles of distributed computing*, pages 240–248.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4599–4610.

Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. 2023. A real-world webagent with planning, long context understanding, and program synthesis. *arXiv preprint arXiv:2307.12856*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625.

Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, et al. 2023. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*.

Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992, Singapore. Association for Computational Linguistics.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019. Pubmedqa: A dataset

for biomedical research 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 2567–2577.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2023. Decomposed prompting: A modular approach for solving complex tasks. In *The Eleventh International Conference on Learning Representations*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

David Lee and Mihalis Yannakakis. 1996. Principles and methods of testing finite state machines-a survey. *Proceedings of the IEEE*, 84(8):1090–1123.

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

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023a. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. 2023b. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*.

Ziyi Liu, Isabelle Lee, Yongkang Du, Soumya Sanyal, and Jieyu Zhao. 2023c. Score: A framework for self-contradictory reasoning evaluation. *arXiv preprint arXiv:2311.09603*.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, Singapore. Association for Computational Linguistics.

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

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 31210–31227. PMLR.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, Toronto, Canada. Association for Computational Linguistics.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. 2023a. A survey on large language model based autonomous agents.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022a. Emergent abilities of large language models. *Transactions on Machine Learning Research*. Survey Certification.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022b. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022c. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2019. Neural text generation with unlikelihood training. In *International Conference on Learning Representations*.

Binfeng Xu, Zhiyuan Peng, Bowen Lei, Subhabrata Mukherjee, Yuchen Liu, and Dongkuan Xu. 2023. Rewoo: Decoupling reasoning from observations for efficient augmented language models. *arXiv preprint arXiv:2305.18323*.

Xin Xu, Yue Liu, Panupong Pasupat, Mehran Kazemi, et al. 2024. In-context learning with retrieved demonstrations for language models: A survey. *arXiv preprint arXiv:2401.11624*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023b. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Da Yin, Faeze Brahman, Abhilasha Ravichander, Khyathi Chandu, Kai-Wei Chang, Yejin Choi, and Bill Yuchen Lin. 2023. Lumos: Learning agents with unified data, modular design, and open-source llms. *arXiv preprint arXiv:2311.05657*.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. 2023. Agenttuning: Enabling generalized agent abilities for llms. *arXiv preprint arXiv:2310.12823*.

A Methodology

A.1 Full Algorithm and Prompts of AMOR

Alg. 2 in the main paper illustrates a general FSM-based reasoning logic, which can be adapted to various agent environments by defining the FSM including the states, modules, etc.

As shown in Alg. 3, AMOR provides an instantiation of the FSM-based reasoning logic for the knowledge-seeking scenarios following the state transition diagram in Fig. 1 in the main paper. We expect to extend this work to more environments in the future.

A.2 Prompts for LLM Modules

Tab. 15, 16, 17 and 18 show the prompts for four LLM modules in AMOR under the "Without Fine-tuning" setting on HotpotQA. They can be converted to the "With Fine-tuning" setting by removing the in-context examples. The prompts for PubMedQA and QASPER are similar.

A.3 Construction of Warm-Up Examples

In this section, we elaborate the pipeline to collect training examples for the warm-up stage of AMOR. Given a sample question Q with annotations of the final answer $\hat{A}$, all sub-queries and answers $\hat{H} = [(\hat{q}_j, \hat{a}_j)]_{j=0}^{J-1}$, and all evidence passages $\hat{E} = [\hat{e}_j]_{j=0}^{J-1}$, where J is the number of necessitated sub-queries of Q, we construct training examples for four LLM modules of AMOR as follows:

- **Decompose**(Q, H)**:** We construct a total of $J+1$ training examples for this module. For each of the J sub-queries, we create an example with the main question Q and the preceding sub-queries and answers $H = \hat{H}_{<j}$ as the input, and the next sub-query $\hat{q}_j$ coupled with the branch token "[NEXT]" as the output (for $j = 0, 1, \ldots, J-1$). Here, $\hat{H}_{<j}$ denotes the sequence containing the first j pairs of sub-queries and their corresponding answers from $\hat{H}$. Additionally, we create one example where the input includes Q and the complete set of sub-queries and answers $H = \hat{H}$, with the branch token "[FINISH]" as the output, indicating the end of the decomposition.
- **Judge**(Q, H, q, d)**:** For this module, the input consists of the main question Q, the previous sub-queries and answers $H = \hat{H}_{<j}$, the current sub-query $q = \hat{q}_j$, and a document snippet d (for $j = 0, 1, \cdots, J-1$). The output is a branch token that classifies the snippet d as either

Algorithm 3 Answering Question Q Using AMOR

Data: AMOR at the initial state $s = s_0$ (Q, H, E); Q: Question; $H = [\,]$: All solved sub-queries and answers; $E = [\,]$: All evidence passages collected by AMOR.
Result: A: Final Answer; R: Reasoning Process.

while True **do**
  if $s = s_0$ **then**
    $y =$ Decompose$(s.Q, s.H)$
    R.append({"state": s, "output": y})
    // Transit to the next state.
    if y starts with "[NEXT]" **then**
      Extract the next sub-query q from y
      $s = s_1(s.Q, s.H, s.E, q)$
    else if y starts with "[FINISH]" **then**
      $s = s_6(s.Q, s.E)$
  else if $s = s_1$ **then**
    $y =$ SearchDoc$(s.q)$
    // Transit to the next state.
    $D, d = [y], y$
    $s = s_2(s.Q, s.H, s.E, s.q, D, d)$
  else if $s = s_2$ **then**
    $y =$ Judge$(s.Q, s.H, s.q, s.d)$
    R.append({"state": s, "output": y})
    // Transit to the next state.
    if y starts with "[IRRELEVANT]" **then**
      $s = s_3(s.Q, s.H, s.E, s.q, s.D)$
    else if y starts with "[RELEVANT]" **then**
      $s = s_4(s.Q, s.H, s.E, s.q, s.D, s.d)$
  else if $s = s_3$ **then**
    $y =$ NextDoc()
    // Transit to the next state.
    if d is NONE **then**
      $H = s.H + [(s.q,$"No Answer"$)]$
      $E = s.E + [s.D[0]]$
      $s = s_0(s.Q, H, E)$
    else
      $D, d = s.D + [y], y$
      $s = s_2(s.Q, s.H, s.E, s.q, D, d)$
  else if $s = s_4$ **then**
    $y =$ SearchPsg$(s.q, s.d)$
    // Transit to the next state.
    $P = y$
    $s = s_5(s.Q, s.H, s.E, s.q, s.D, P)$
  else if $s = s_5$ **then**
    $y =$ Answer(Q, H, q, P)
    R.append({"state": s, "output": y})
    // Transit to the next state.
    if o starts with "[UNANSWERABLE]" **then**
      $s = s_3(s.Q, s.H, s.E, s.q, s.D)$
    else if o starts with "[ANSWERABLE]" **then**
      Extract the answer a and the evidence p from y
      $H = s.H + [s.q, a]$
      $E = s.E + [e]$
      $s = s_0(s.Q, H, E)$
  else if $s = s_6$ **then**
    $y =$Complete$(s.Q, s.E)$
    R.append({"state": s, "output": y})
    $A = y$ // Reach the final state.
    break
return A, R

Module m	Output y	Gold Process Feedback f_{human}	ACC_f
Decompose(Q, H)	[NEXT] q [FINISH]	"right", if q is a reasonable sub-query for solving Q; "wrong"; otherwise. "right", if there are no more sub-queries required; "wrong", otherwise.	1, if $f = f_{\text{human}}$, 0, otherwise.
Judge(Q, H, q, d)	[RELEVANT] [IRRELEVANT]	"[RELEVANT]", if d is relevant with q; "[IRRELEVANT]", otherwise.	1, if $f = f_{\text{human}}$; 0, otherwise.
Answer(Q, H, q, P)	[ANSWERABLE] a e [UNANSWERABLE]	"right", if a is the correct answer to q evidenced by e; "wrong", otherwise "right", if q can not be answered based on P; "wrong", otherwise	1, if $f = f_{\text{human}}$; 0, otherwise.
Complete(Q, E)	A	"right", if E evidence that Q can be answered by A; $\hat{A}$, else if E evidence that Q can be answered by $\hat{A}$; "wrong", otherwise.	1, if $f = f_{\text{human}}$ or $f = \hat{A}$; 0, otherwise.

Tab. 9: The process feedback annotation strategy with humans for different LLM modules, as well as the method to calculate the accuracy ACC_f of a piece of silver feedback f.

"[RELEVANT]" or "[IRRELEVANT]" in relation to the current sub-query $\hat{q}_j$. We consider three scenarios for the document snippet d: (1) When d is the gold evidence passage $\hat{e}_j$, the output is "[RELEVANT]". (2) When d is a passage from a different document from $\hat{e}_j$, it is marked as "[IRRELEVANT]". We obtain this type of snippet, denoted as d_j, by using $\hat{q}_j$ as the query in SearchDoc, ensuring it originates from a distinct document compared to $\hat{e}_j$. (3) When d is a passage from the same document as $\hat{e}_j$ but is not $\hat{e}_j$ itself, it is deemed "[RELEVANT]". We acquire such snippets by invoking SearchPsg with $\hat{q}_j$ to retrieve passages from the same document as $\hat{e}_j$, excluding $\hat{e}_j$ from the results. We refer to this set of passages as P^-, considering each of them relevant to $\hat{q}_j$. These varied document snippet scenarios are designed to train the module to discern the relevance of a query to a document based solely on portions of the document content.

- **Answer**(Q, H, q, P)**.** Similar to the Judge module, the input for this module comprises the main question Q, the previous sub-queries and answers $H = \hat{H}_{<j}$, the current sub-query $q = \hat{q}_j$, and a set of passages P from the same document. The output is either the branch token [UNANSWERABLE]" or a combination of the branch token [ANSWERABLE]", the corresponding answer $\hat{a}_j$, and evidence passage $\hat{e}_j$. We consider two scenarios for P: (1) When P does not include $\hat{e}_j$, indicating that the sub-query $\hat{q}_j$ cannot be answered, the output is "[UNANSWERABLE]". Here, P is set to the previously mentioned set P^-. (2) When P includes $\hat{e}_j$, suggesting that $\hat{q}_j$ is answerable, we create P by replacing a random passage in P^- with $\hat{e}_j$. For both scenarios, we present the passages to the module in random order when constructing training examples.
- **Complete**(Q, E)**.** We construct one training example for this module by setting the input to the main question Q and gold evidence $\hat{E}$ and the output to the final answer $\hat{A}$.

After generating examples from the warm-up datasets using the aforementioned pipeline, we randomly select a specified number of examples, as detailed in Tab. 2 of the main paper. This random sampling aims to ensure a balanced representation of the various modules and branch tokens in the final dataset.

B Experiments

B.1 Tool Modules

We implement both SearchDoc and SearchPsg by adapting Contriever. Given a query, SearchDoc first uses Contriver to retrieve a number of passages from a specific knowledge base and only retains the most relevant passage from each document to serve as the document's representative snippet. Then, SearchDoc returns the top one document snippet and NextDoc can return at most nine more snippets from the remaining ones. On the other hand, SearchPsg returns the top three passages within a given document retrieved using Contriever.

The operation of these tools mirrors the hierarchical interaction paradigm that humans use with search engines (Yao et al., 2023b; Yin et al., 2023): they first identify a relevant document based on short snippets and then refine the search results by focusing within the document.

B.2 Adaptation & Evaluation Datasets

We describe how we process the datasets as follows: **(1) HotpotQA**: Each document is a Wikipedia article. Since the original test set is hidden, we randomly sample 500 examples from the original validation set for evaluation and split the training set for adaptation fine-tuning and validation. **(2) PubMedQA** (Jin et al., 2019): We follow the official split. And we only remain examples whose

Method	Decompose	Judge	Answer	Complete
AMOR$_{\text{Process}}$	81.3	95.2	84.4	82.0

Tab. 10: Accuracy of the silver feedback for four LLM modules based on L-7B.

answers are "yes" or "no" and discard those labeled "maybe." **(3) QASPER** (Dasigi et al., 2021): For each question, we regard the corresponding paper as a knowledge base and each section of the paper as a document with the section name as the title (e.g., "Experiments::Datasets") including several passages. Although many LLMs support context longer than the average paper length of 7k tokens, we focus on testing the ability of language agents to seek and utilize information in this work. We exclude questions that are labeled "unanswerable." Since the original test set is also hidden, we use the original validation set for evaluation and redivide the training set for training and validation.

B.3 Reasoning Process Assessment

To investigate the extent to which the adaptation stage enhances AMOR's reasoning process, we conducted a human study with one NLP expert using the HotpotQA test set, Tab. 9 demonstrates the protocol for annotating the gold feedback f_{human} and how we calculate the accuracy of the automatic silver feedback f in Tab. 3, denoted as ACC$_f$. Based on f_{human}, we measured the accuracy of each LLM module's output y (denoted as ACC$_m$) as follows:

$$\text{ACC}_m = \begin{cases} 1 & \text{if } f_{\text{human}} = \text{“right”}, \\ 1 & \text{if } f_{\text{human}} \text{ is a refinement of } y \\ & \quad \text{and } f_{\text{human}} = y, \\ 0 & \text{if } f_{\text{human}} = \text{“wrong”}, \\ 0 & \text{if } f_{\text{human}} \text{ is a refinement of } y \\ & \quad \text{and } f_{\text{human}} \neq y. \end{cases} \quad (4)$$

The accuracy of the reasoning process ACC$_m$ has been discussed in Tab. 8 of the main paper. Furthermore, Tab. 10 presents the accuracy of the silver feedback ACC$_f$ for AMOR$_{\text{Process}}$. The silver feedback achieves an ACC$_f$ above 80% for all modules, lending credibility to the use of silver feedback in the adaptation experiments.

B.4 Multi-Round Adaptation

In the main paper, we set $I = 1$ in Alg. 2 for all experiments, which means that all exploratory instances in the adaptation stage are induced by the warm-up policy AMOR$_{\text{WFT}}$. We call this setting "single-round adaptation." We are curious about how multi-round adaptation influences the performance of AMOR by adjusting I. For the i-th iteration ($i = 1, 2, \cdots, I$), we denote the initial parameter of AMOR as θ_{i-1}, which is used to explore over a set of input questions and is updated to θ_i after exploitation using these exploratory instances. AMOR$_{\theta_0}$ is exactly AMOR$_{\text{WFT}}$. During different iterations, we can provide either the same or different questions for AMOR to explore over. The case with the same set of questions is used to simulate an exploration-limited scenario. Note that in this case, the exploratory instances with the same questions are still different due to the ever-changing policy leading to different outputs.

Metric	Same Questions			Different Questions		
	θ_1	θ_2	θ_3	θ_1	θ_2	θ_3
EM	41.4	41.0	41.4	41.4	40.4	40.0
F1	50.9	49.4	50.7	50.9	50.5	49.6

Tab. 11: Performance of AMOR parameterized by θ_i during multi-round adaptation. In the i-th iteration ($i = 0, 1, 2$), AMOR$_{\theta_i}$ is used to explore over the same set of questions or different ones and then is updated to AMOR$_{\theta_{i+1}}$ based on the exploratory instances.

Method	Base LLM	HotpotQA	PubMedQA	QASPER
ReAct	**GPT-4**	-	19.0k	25.3k
AMOR$_{\text{w/o FT}}$	**GPT-4**	11k	7.7k	6.3k
AgentLM	**L-7B**	-	7.0k	8.9k
AMOR$_{\text{Process}}$	**L-7B**	4.3k	2.6k	2.1k

Tab. 12: Average LLM token usage of different agents.

Tab. 11 shows the performance of AMOR under the multi-round adaptation setting with $I = 3$. We find that the performance is almost unchanged whether using the same or different input questions for each adaptation round. This result suggests that one iteration may be sufficient for the adaptation fine-tuning stage in our study.

B.5 Token Efficiency

Language agents interact with environments to solve problems through frequent calls of LLMs, leading to huge costs in terms of token consumption. Building agents with minimal token usage is essential for curbing deployment costs (Xu et al., 2023).

Table 12 displays the average number of tokens used by AMOR and ReAct-style agents to answer

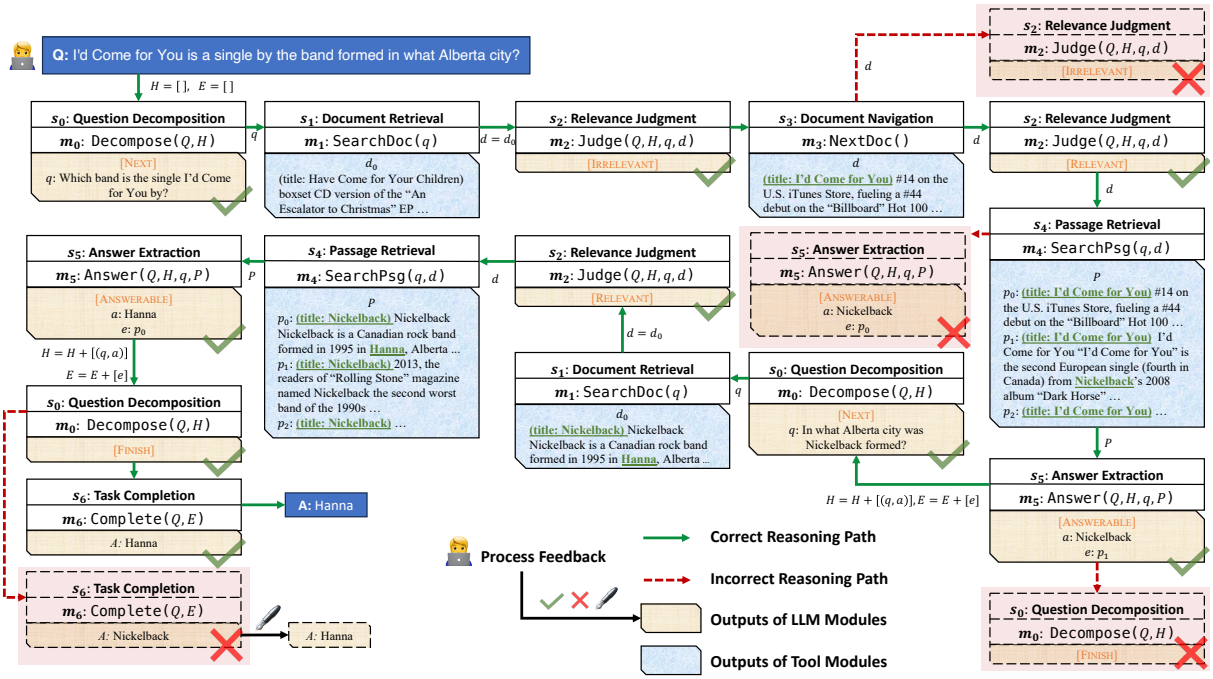

Fig. 4: An example demonstrating how AMOR$_{\text{Process}}$ answers a complex question from HotpotQA. Users are allowed to provide direct process feedback to drive the evolution of the agent.

a question, accounting for both input and output tokens. ReAct-style agents, lacking explicit modeling of inter-step dependencies, require the inclusion of all preceding information in the input for each step. This often results in undesired redundancy. In contrast, AMOR consumes significantly fewer tokens with each module relying only on essential historical information, which highlights the token efficiency of its architecture.

B.6 Flexibility

FSM-based reasoning logic is flexible in facilitating targeted enhancements of specific modules and easily accommodating new tools. We conduct two experiments as follows on HotpotQA to demonstrate the flexibility of AMOR, with results shown in Tab. 14.

(1) Targeted Fine-tuning. Tab. 8 reveals that the Complete module of AMOR$_{\text{Process}}$ still falls short in performance, achieving only $\sim 50\%$ accuracy. We construct 6k examples for the module from the original training set of HotpotQA by treating the final answer $\hat{A}$ as input, and the question Q and evidence passages $\hat{E}$ as output, and then fine-tune the L-7B model on the data. Tab. 14 shows the performance gains when substituting the original Complete module in AMOR$_{\text{Process}}$ with this individually fine-tuned L-7B model.

Question: I'd Come for You is a single by the band formed in what Alberta city?

Thought 1: We need to find the Alberta city where the band formed. Let's search for information about the band and their formation.
Action 1: SearchDoc(Query="Information about the band formed that released the single 'I'd Come for You'. ")
Observation 1: (title: I'd Come for You) ... #14 on the U.S. iTunes Store, fueling a #44 debut on the "Billboard" Hot 100 ...

Thought 2: No relevant information is found. Let's move to the next document.
Action 2: NextDoc()
Observation 2: (title: Coming for You) Coming for You "Coming for You" is a song by the American punk rock band the ...

Thought 3: There is still no relevant information found. Let's try searching for information about the single.
Action 3: SearchDoc(Query="Information about the single 'I'd Come for You'.")
Observation 3: (title: I'd Come for You) ... #14 on the U.S. iTunes Store, fueling a #44 debut on the "Billboard" Hot 100 ...

Thought 4: No relevant information found. Let's try searching for the lyrics of the single.
Action 4: SearchDoc(Query="Lyrics of the single 'I'd Come for You'.")
Observation 4: (title: I'd Come for You) I'd Come for You "I'd Come for You" is the second European single (fourth in Canada) from Nickelback's 2008 album "Dark Horse" ...

Thought 5: The relevant information can be found in the document. The single "I'd Come for You" is from Nickelback's album "Dark Horse" and was released on March 23, 2009. The band Nickelback was formed in Alberta, without specific information about the city. We do not have enough information to answer the task perfectly. Please try another question.
Action 5: Finish(Result="Not enough information to answer the question perfectly")

Fig. 5: A failure case of ReAct (built upon GPT-3.5) when answering a complex question from HotpotQA by reasoning over retrieved knowledge. The wrong steps are highlighted in red while the right ones are in green.

(2) Accommodating New Tools. Numerous studies have demonstrated the benefits of retrieval-based in-context learning, where a retriever selectively curates tailored demonstrations for each specific input (Xu et al., 2024). We implement this by inserting a new state s_6', named "Demonstration Retrieval," before the final state s_6 shown in the state transition diagram in Fig. 1, making AMOR reach s_6' when Decompose outputs "[FINISH]" at state

Question: Were the bands Skin Yard and Ostava from the U.S.?

Decompose: [NEXT] Which band has Skin Yard been part in?
SearchDoc: (title: Skin Yard) States of America has since reunited, with Finn still drumming · · ·
Judge: [RELEVANT]
SearchPsg: [1] (title: Skin Yard) States of America has since reunited, with Finn still drumming · · ·
[2] (title: Skin Yard) Skin Yard Skin Yard was an American grunge band from Seattle, Washington, who were active from 1985 to 1993 · · ·
[3] (title: Skin Yard) · · · He produced several albums by the grunge bands Soundgarden (a band including original Skin Yard drummer Matt Cameron) · · ·
Answer: [ANSWERABLE] Answer: Soundgarden; Relevant Passage ID: [3]
Decompose: [NEXT] Which band has Ostava been part in?
SearchDoc: (title: Ostava) Ostava Ostava is an alternative rock band from Bulgaria · · ·
Judge: [RELEVANT]
SearchPsg: [1] (title: Ostava) Ostava Ostava is an alternative rock band from Bulgaria · · ·
[2] (title: Ostava) of live album. In summer 2006, Ostava supported Ladytron at their second concert in this time at Sunny Beach, Bulgaria · · ·
[3] (title: Ostava) never officially released a cover version, they do play covers occasionally · · ·
Answer: [ANSWERABLE] Answer: Ladytron; Relevant Passage ID: [2]
Decompose: [FINISH]
Complete: no

Tab. 13: A case where AMOR$_{\text{WFT}}$ answers a question from HotpotQA with the right final answer and evidence but a wrong reasoning process. Each line denotes a reasoning step and shows the output of the corresponding module. The steps highlighted in green are right while those in red are wrong.

Method	EM	F1
AMOR$_{\text{Process}}$	41.4	50.9
+Targeted Fine-tuning of Complete	43.4	51.9
+Additional Tool SearchDemo	43.8	52.7

Tab. 14: AMOR can be enhanced through targeted fine-tuning and flexibly accommodate additional tools. All results are based on L-7B.

s_0. The new state s_6' holds two variables, including the main question Q and all collected evidence E, and employs a tool module SearchDemo to retrieve the top K similar questions to Q from an external demonstration memory, along with their answers and evidence, collectively noted as $\mathcal{K} = [Q_k, \hat{A}_k, \hat{E}_k]_{k=1}^{K}$. Subsequently, at state s_6, the Complete module takes $\mathcal{K}$ as the in-context examples, helping generate the final answer A given Q and E. We use the HotpotQA training set as our demonstration memory and employ Contriever-MS MARCO (Izacard et al., 2022) to implement the SearchDemo module, setting K to 5. We fine-tune the L-7B model on the training set to act as the Complete module while ensuring that the demonstration does not include the target question. As Table14 indicates, this integration of such an additional tool further improves AMOR$_{\text{Process}}$ with targeted fine-tuning.

Additionally, AMOR's reasoning logic can be easily expanded from single-path to multi-path reasoning, akin to the approaches used in Self-Consistency (Wang et al., 2023b), ToT (Yao et al., 2023a), and GoT (Besta et al., 2023). This can be achieved by generating multiple outputs within specific modules and incorporating modules that synthesize these multi-path results. Consequently, we advocate for the adoption of the FSM paradigm in the design of future agents. This framework offers the dual benefits of flexibility and the capacity to adapt agents based on process feedback.

B.7 Case Study

We demonstrate the advantages of the FSM-based reasoning logic and process feedback mechanism through the comparison between AMOR$_{\text{Process}}$ and ReAct in Fig. 4 and 5, respectively. We observe that ReAct without explicit reasoning logic constraints fails to decompose the question and terminates retrieval prematurely in "Thought/Action 5." Besides, ReAct also mixes right and wrong steps in "Thought 2/4/5," making it challenging for users to critique and improve the agent in a targeted manner. In contrast, AMOR successfully answers the question with a controllable reasoning logic and allows direct process feedback to drive the evolution.

Additionally, Table 13 shows a case where AMOR$_{\text{WFT}}$ correctly answers a question with the right evidence, yet employs a wrong reasoning process. This underscores the potential unreliability of using outcome feedback to judge the correctness of the reasoning process and the necessity of employing process feedback for adapting agents to specific environments.

Decompose(Q, H)

Please continue to decompose the provided main question into answerable sub-queries following previously already solved sub-queries. There are two cases as follows:
(1) [Next] If the question requires further decomposition: Identify and output the next logical sub-query that must be addressed in order to progress towards answering the main question.
(2) [Finish] It means the question does not require further decomposition and can be answered as is.

HERE ARE SEVERAL EXAMPLES:
====Examples Start====
(1) Main Question: What U.S Highway gives access to Zilpo Road, and is also known as Midland Trail?
Output: [Next] How can Zilpo Road be accessed?

(2) Main Question: Which magazine was started first Arthur's Magazine or First for Women?
Solved Sub-Queries:
1. Q: When was Arthur's Magazine started? A: 1844-1846
Output: [Next] When was First for Women magazine started?

(3) Main Question: Which magazine was started first Arthur's Magazine or First for Women?
Solved Sub-Queries:
1. Q: When was Arthur's Magazine started? A: 1844-1846
2. Q: When was First for Women magazine started? A: 1989
Output: [Finish]
====Examples End====

Now Please Complete the Following Task. Please ensure that each sub-query is specific enough to understand in isolation.
Main Question: $\{Q\}\{H'\}$ $\{\%H'$ is a formatted string representing the solved sub-queries and answers constructed from $H.\%\}$
Output:

Tab. 15: Prompt for the Decompose module for HotpotQA.

Judge(Q, H, q, d)

Given a sub-query derived from the main question and a document snippet with its title, please assess whether the document is potentially relevant to the sub-query based on the title and shown content of the document. Assign one of the following two categories:
(1) [Relevant]: Choose this category if the document is relevant to the sub-query.
(2) [Irrelevant]: Choose this category if the document is irrelevant to the sub-query.

HERE ARE SEVERAL EXAMPLES:
====Examples Start====
(1) Main Question: Which magazine was started first Arthur's Magazine or First for Women?
Next Sub-Query: When was Arthur's Magazine started?
Document Snippet: (title: Arthur's Magazine) Arthur's Magazine Arthur's Magazine (1844-1846) was an $\cdots$
Output: [Relevant]

(2) Main Question: Which magazine was started first Arthur's Magazine or First for Women?
Solved Sub-Queries:
1. Q: When was Arthur's Magazine started? A: 1844-1846
Next Sub-Query: When was First for Women magazine started?
Document Snippet: (title: History of women's magazines) In 1693 the first issue of the first women's magazine in Britain $\cdots$
Output: [Irrelevant]

(3) Main Question: What U.S Highway gives access to Zilpo Road, and is also known as Midland Trail?
Next Sub-Query: How can Zilpo Road be accessed?
Document Snippet: (title: Zilpo Road) constructed on the Licking River by the Army Corps of Engineers. $\cdots$
Output: [Relevant]
====Examples End====

Now Please Complete the Following Task:
Main Question: $\{Q\}\{H'\}$ $\{\%H'$ is a formatted string representing the solved sub-queries and answers constructed from $H.\%\}$
Next Sub-Query: $\{q\}$
Document Snippet: d
Output:

Tab. 16: Prompt for the Judge module for HotpotQA.

Answer(Q, H, q, P)

Please assess whether the sub-query derived from the main question can be answered using the information from the provided passages. Your evaluation should categorize the sufficiency of the information in the passages with respect to the sub-query. Assign one of the following three categories:
(1) [Unanswerable]: Choose this category if the given passages do not contain information to answer it directly.
(2) [Answerable]: Use this category if one of the given passages contains sufficient information to directly answer the sub-query. Provide a clear and concise answer to the sub-query, and the ID of the the corresponding passage.

HERE ARE SEVERAL EXAMPLES:
====Examples Start====
(1) Main Question: Which magazine was started first Arthur's Magazine or First for Women?
Solved Sub-Queries:
1. Q: When was First for Women magazine started? A: 1989
Next Sub-Query: When was Arthur's Magazine started?
Passages: [1] (title: Arthur's Magazine) He was also the author of dozens $\cdots$
[2] (title: Arthur's Magazine) Arthur's Magazine Arthur's Magazine (1844-1846) was an $\cdots$
[3] (title: Arthur's Magazine) The articles were widely reprinted and helped fuel $\cdots$
Output: [Answerable] Answer: 1844-1846; Relevant Passage ID: [2]

(2) Main Question: What U.S Highway gives access to Zilpo Road, and is also known as Midland Trail?
Next Sub-Query: How can Zilpo Road be accessed?
Passages: [1] (title: Zilpo Road) the city which transports people in and out of the city $\cdots$
[2] (title: Zilpo Road) Grand Terrace. Access provides public transportation services $\cdots$
[3] (title: Zilpo Road) On the other side of the lake is the 700-acre (280 ha) $\cdots$
Output: [Unanswerable]
====Examples End====

Now Please Complete the Following Task:
Main Question: {Q}{H'} {$\%H'$ is a formatted string representing the solved sub-queries and answers constructed from H.$\%$}
Next Sub-Query: {q}
Passages: {P}
Output:

Tab. 17: Prompt for the Answer module for HotpotQA.

Complete(Q, E)

Answer the question ONLY based on the provided passages. Your output should be "yes/no" or a short entity.

HERE ARE SEVERAL EXAMPLES:
====Examples Start====
(1) Question: Which magazine was started first Arthur's Magazine or First for Women?
Passages: [1] (title: Arthur's Magazine) Arthur's Magazine Arthur's Magazine (1844-1846) was an $\cdots$
[2] (title: First for Women) First for Women $\cdots$ was started in 1989 $\cdots$
Output: Arthur's Magazine

(2) Question: What U.S Highway gives access to Zilpo Road, and is also known as Midland Trail?
Passages: [1] (title: Zilpo Road) Zilpo Road $\cdots$ can be accessed by Kentucky Route 211 (KY 2112) $\cdots$
[2] (title: Morehead, Kentucky) Morehead is a home rule-class city[5] located along US 60 (the historic Midland Trail) $\cdots$
Output: US 60
====Examples End====

Question: {Q}
Passages: {E'} {$\%E'$ is a formatted string representing all evidence passages constructed from E.$\%$}
Output:

Tab. 18: Prompt for the Complete module for HotpotQA.