A Metacognitive Architecture for Correcting LLM Errors in AI Agents

Jisu Kim, Mahimul Islam, Ashok Goel
Design Intelligence Laboratory
Georgia Institute of Technology
jisu.kim@gatech.edu, mahimul@gatech.edu, ashok.goel@cc.gatech.edu

Abstract

The ability to correct mistakes and adapt to users' changing needs is critical for AI agents to remain robust and trustworthy. LLM-based agents are inherently prone to errors like hallucinations and misinterpretations. We observed this problem in SAMI, an AI social agent deployed in Georgia Tech's OMSCS program for ten semesters (11,000+ users). During deployment, users frequently asked the agent to revise its knowledge base, both to correct LLM-induced errors and to update their information. To address this need, we introduce a two-level metacognitive self-adaptation architecture that integrates knowledge-based AI (KBAI) with LLMs. The architecture comprises a cognitive layer that performs the agent's core tasks, and a metacognitive layer that introspects on the cognitive layer's process using a Task–Method–Knowledge (TMK) model of the agent. The metacognitive layer identifies the task that needs revision, updates the knowledge base, and communicates the revision process to the user.

Introduction

Artificial intelligence (AI) agents often face situations in which revising their knowledge and outputs is crucial. For example, agents that use large language models (LLMs) such as ChatGPT are inherently prone to errors, including hallucinations and misinterpretations (Bang et al. 2023; Ji et al. 2023). Such mistakes erode trust in human–AI interactions and harm perceptions of the agent's intelligence and likability (Honig and Oron-Gilad 2018; Lee et al. 2024; Salem et al. 2015).

SAMI is an AI social agent deployed in Georgia Tech's Online Master of Science in Computer Science (OMSCS) program for ten semesters, serving over 11,000 users. It recommends social connections between students based on shared interests or characteristics inferred from their online posts (Kakar et al. 2024). SAMI uses ChatGPT to process student posts and generate recommendations and responses. This reliance on ChatGPT can lead to incorrect or misleading outputs. A prior user-perception study on SAMI (Wang 2024), however, found that providing revisions to these errors and transparently communicating the revision process can improve users' perceptions of the agent even after seeing such errors (Ashktorab et al. 2019).

Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

A promising approach to enabling AI agents to adapt to user feedback and communicate this process transparently is to equip them with a metacognitive architecture. Metacognition is the process of "reasoning about one's own reasoning" (Cox 2005). It provides a higher-level mechanism that enables AI agents to reflect on and adapt their behavior (Cox and Raja 2007). Most prior work on metacognition in AI has focused on building human-like agents or improving task performance (Ganapini et al. 2023; Schmill et al. 2008). Leveraging metacognition to correct LLM-induced errors in AI agents and to improve the transparency of their behavior to users remains underexplored (Wang 2024).

To address this gap, we introduce a metacognitive self-adaptation architecture. We first categorize revision-need types by analyzing 32 GitHub issues observed during real deployments. We then implement a two-level self-adaptation architecture that integrates knowledge-based AI (KBAI) with LLMs. In this architecture, a cognitive layer performs the agent's core tasks, and a metacognitive layer uses the Task–Method–Knowledge (TMK) model to introspect on the cognitive layer. The metacognitive layer (1) localizes the task requiring revision, (2) updates the knowledge base, and (3) communicates the revision process back to the user as a step-by-step explanation. We evaluate the architecture using 20 feedback cases drawn from real student data and demonstrate a path to deployment in the OMSCS program.

We summarize our main contributions as follows.

- A two-level metacognitive self-adaptation architecture that localizes and corrects LLM-induced errors in deployed AI agents.
- An integration of KBAI and LLMs that combines TMK and a knowledge graph with ChatGPT to support robust and interpretable adaptation.

Related Work

Metacognition in AI agents has been studied as a means to monitor, explain, and improve agent behavior. Examples include architectures for self-diagnosis that generate self-expectations to monitor and explain failures (Schmill et al. 2008), and deliberative agents that arbitrate between thinking fast (System 1) and thinking slow (System 2) (Ganapini et al. 2023).

TMK provides a structured, interpretable self-model of an AI agent's internal processes (Goel and Rugaber 2017; Murdock and Goel 2008). TMK is more expressive than Hierarchical Task Networks (Erol, Hendler, and Nau 1994; Nau et al. 2003) because it captures expectations of tasks and subtasks, supporting both prediction and explanation (Hoang, Lee-Urban, and Muñoz Avila 2005; Lee-Urban and Muñoz-Avila 2006). Prior work has used TMK for self-adaptation of an agent's design (Goel and Rugaber 2015, 2017) and, more recently, for self-explanation in SAMI (Basappa et al. 2024).

Since the advent of LLMs such as ChatGPT, researchers have explored several approaches to combining KBAI with LLMs. These include infusing symbolic knowledge into neural networks (Gaur and Sheth 2024), using LLMs to construct knowledge representations that are then used for symbolic reasoning (Kirk et al. 2024; Lawley and Maclellan 2024), and modeling LLMs as System 1 and KBAI as System 2 (Ganapini et al. 2023).

This paper builds on this earlier work in three ways. First, it adapts metacognition to a deployed AI agent that repairs LLM-induced errors in response to user feedback. Second, it uses TMK to localize where revisions are needed within the agent's internal process. Third, it integrates KBAI, specifically TMK and a knowledge graph, with LLMs such as ChatGPT to revise the knowledge base and generate explanations for users.

Problem Background

To ground these ideas in a deployed agent, we describe SAMI and the types of errors observed in practice. SAMI is an AI social agent designed to mitigate the social challenges faced by online students. It was developed through co-design workshops with students in Georgia Tech's OMSCS program to support their need for meaningful social connections (Wang, Jing, and Goel 2022). SAMI integrates into course discussion forums, specifically Ed Discussion (Ed). It uses ChatGPT to extract entities from students' posts (e.g., locations, interests) and to generate personalized recommendation messages. We have deployed SAMI for ten semesters in courses such as Human–Computer Interaction and Knowledge-Based AI, serving over 11,000 students (Kakar et al. 2024).

During deployment, students frequently requested revisions to SAMI's knowledge base, either to correct agent-generated errors or to update their own information. We categorized 32 GitHub issues reported across deployments into revision-need types (Table 1). We addressed the issue types marked * through software engineering. However, the remaining open issue types depend on ChatGPT and remain unresolved. Although ChatGPT performs competitively with named-entity recognition methods, it remains prone to errors.

A prior empirical user-perception study (Wang 2024) further showed that such errors undermine users' trust in the agent and discourage continued use. However, when the agent revises its behavior based on user feedback and transparently communicates how it adapted, users' perceptions of the AI improve significantly. These observations underscore the need for a structured mechanism that enables the agent to identify, revise, and explain errors reliably.

A Metacognitive Self-Adaptation Architecture

We present a metacognitive self-adaptation architecture comprising two processing layers: a cognitive layer (Level 1) and a metacognitive layer (Level 2). At Level 1 (the cognitive layer), the agent performs its core tasks. At Level 2 (the metacognitive layer), it introspects on Level 1 to identify the task requiring revision, update the knowledge base, and generate a revision message in response to user feedback (Figure 1). In the rest of this section, we describe how this architecture is instantiated in SAMI.

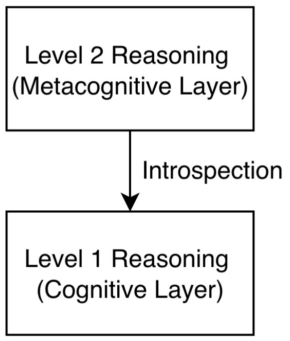

Figure 1: Overview of the two-level metacognitive self-adaptation architecture. Level 1 executes the agent's primary tasks, and Level 2 introspects on Level 1 to localize and revise errors.

Level 1 Reasoning

At Level 1, the agent generates initial social recommendations based on users' introduction posts. The agent first classifies the post type using LangChain[1] and ChatGPT[2]. If the post is an introduction, the agent uses ChatGPT to extract entities such as hobbies, locations, and academic interests. The extracted information is then stored in the knowledge base as a knowledge graph. Using this knowledge base, the agent applies a matchmaking algorithm to identify users with shared attributes and then generates personalized recommendation messages using ChatGPT (Kakar et al. 2024).

Task–Method–Knowledge Representation

To support Level 2 introspection over Level 1, we use TMK to represent the agent's Level 1 reasoning process. TMK provides a structured and interpretable self-model of the agent's internal processes. It does so by encoding three components: Tasks denote the goals the agent tries to achieve, Methods specify how these tasks are executed, and Knowledge refers to the information the agent uses (Goel and Rugaber 2015). TMK enables self-explanation, allowing the agent to answer questions such as "*What kind of data do you*

[1]https://docs.langchain.com/

[2]We used OpenAI's gpt-4o-mini model.

Task	Type	Definition	Example Agent Behavior
Identify and Extract Entities from Ed Post	Hallucination	Agent extracts entities not present in the post.	Agent extracts “Atlanta” as the primary location, even though the user did not mention it.
	Omission	Agent fails to extract entities actually present in the post.	Agent misses the hobby “hiking” even though the user mentioned it.
	Misinterpretation	Agent extracts entities correctly but fails to infer the correct contextual meaning.	Agent interprets “New York” as the current primary location when it was mentioned as a previous location.
	User-Initiated Update	User requests a change to the knowledge base due to external changes or inaccuracies in the post.	User requests an update of the primary location from “Chicago” to “Seattle” after a recent move.
Match Users	Mismatch*	Agent provides incorrect or unhelpful matches.	Agent matches users solely on enrollment in the same course, which applies to all users in the same discussion forum.
Generate Responses	Hallucination*	Agent includes entities that are not present in the knowledge base in its response.	Agent describes “Knowledge-Based AI” as a shared course, even though it is not in the knowledge base.
	Misinterpretation*	Agent misinterprets entities in the knowledge base.	Agent marks shared locations as “Unknown” when the shared primary locations are absent from the knowledge base.
	Misattribution*	Agent generates a correct response, but associates it with the wrong user.	Agent describes the user who wrote the post instead of the matched user.

Table 1: Categorization of observed revision-need types in SAMI during deployments across ten semesters at Georgia Tech’s OMSCS program. Each entry is classified according to the relevant Task, Definition, and Example Agent Behavior. Types marked * were successfully resolved through traditional software re-engineering. The remaining open issue types arise from errors inherent to ChatGPT as a zero-shot learner.

learn?” and ”*How do you find matches for users?*” (Basappa et al. 2024).

For self-adaptation, we focus on the Task model because it decomposes the agent’s Level 1 process into interpretable units (Table 2). This expressiveness provides the structural basis for localizing where a revision is needed. Each Level 1 mistake can be traced to a specific task in the Task model. For example, the agent may extract the wrong entity, make errors in the matchmaking algorithm, or generate incorrect responses (Table 1). We constructed and refined SAMI’s Task model across deployments to reflect updates to its working mechanism.

Level 2 Reasoning

At Level 2, the metacognitive layer introspects on Level 1 using the TMK. When a user’s post is classified as feedback requesting a revision, the agent executes Level 2 in two stages. Task Localization identifies the task requiring revision, and Knowledge Revision updates the knowledge base (Figure 2). Each stage generates intermediate natural-language messages describing the agent’s actions and reasoning. These messages are later combined into a single revision explanation presented to the user.

Schema Field	Example
name	Identify and Extract Entities from Ed Post
description	Extract entities (locations, names, details) from #connectme posts
inputs	Sentence from Ed forum post
outputs	Extracted entities
method	ChatGPT-based Named Entity Recognition (NER)
parent	NLPModule.prepare_features

Table 2: Schema for one of the tasks in the TMK. The Task model provides the structure for the agent to determine which part of Level 1 requires revision.

Task Localization The agent localizes the task requiring revision using Algorithm 1. First, it extracts task-relevant entities from the user feedback. It then uses the FAISS library[3] to compute similarity between the feedback and task descriptions in the TMK, selecting the most relevant task. Next, the agent retrieves the user’s data from the knowl-

[3]https://faiss.ai/index.html

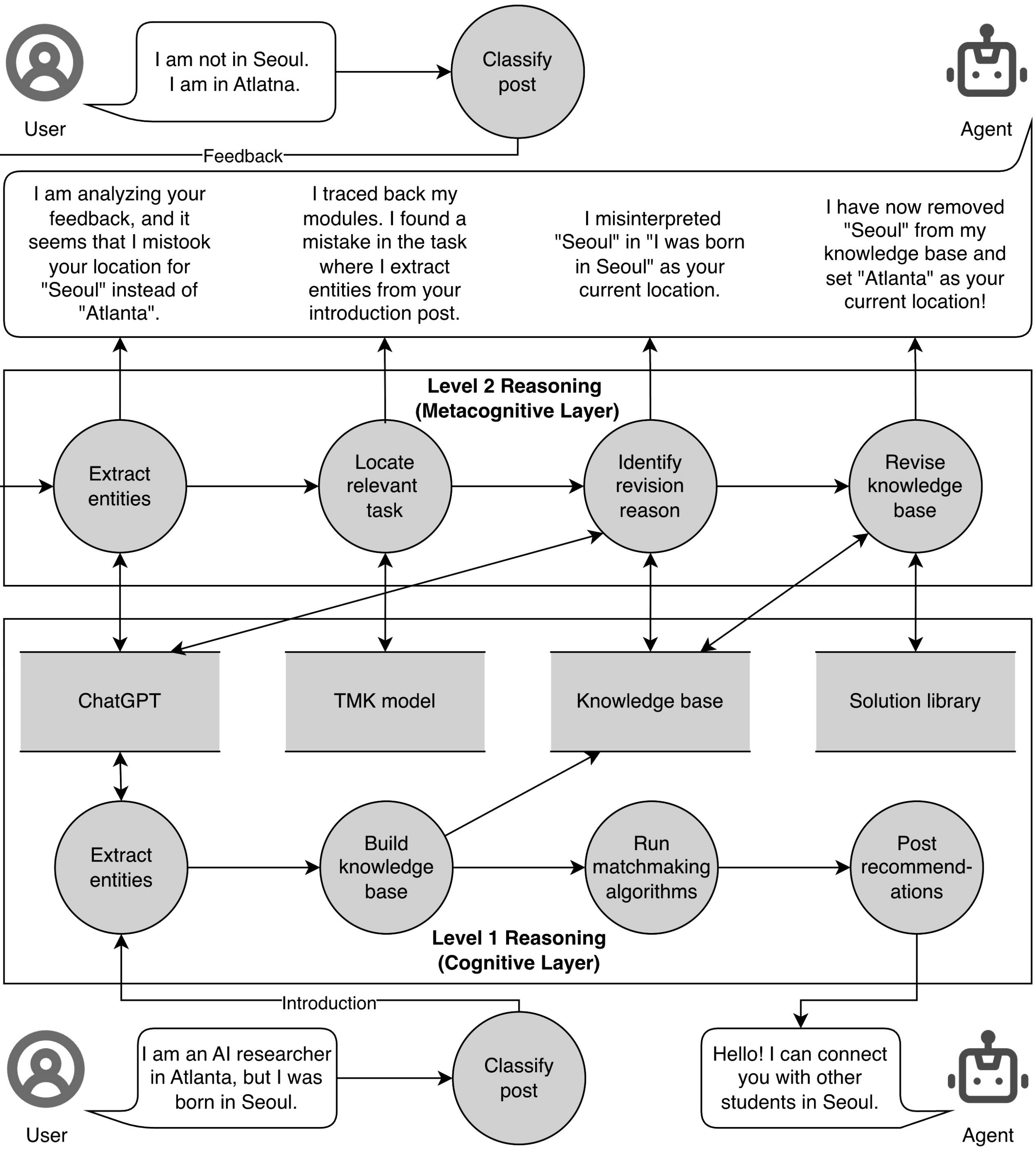
User
I am not in Seoul. I am in Atlatna.
Classify post
Agent
Feedback
I am analyzing your feedback, and it seems that I mistook your location for "Seoul" instead of "Atlanta".
I traced back my modules. I found a mistake in the task where I extract entities from your introduction post.
I misinterpreted "Seoul" in "I was born in Seoul" as your current location.
I have now removed "Seoul" from my knowledge base and set "Atlanta" as your current location!
Level 2 Reasoning (Metacognitive Layer)
Extract entities
Locate relevant task
Identify revision reason
Revise knowledge base
ChatGPT
TMK model
Knowledge base
Solution library
Extract entities
Build knowledge base
Run matchmaking algorithms
Post recommend-ations
Level 1 Reasoning (Cognitive Layer)
Introduction
User
I am an AI researcher in Atlanta, but I was born in Seoul.
Classify post
Hello! I can connect you with other students in Seoul.
Agent

Figure 2: Data-flow diagram of SAMI's metacognitive self-adaptation process, integrating KBAI (TMK, knowledge graph, solution library) with LLMs (ChatGPT). The bottom flow (Level 1) shows how the agent extracts entities with ChatGPT, builds a knowledge base, and generates social recommendations. The top flow (Level 2) shows how user feedback triggers introspection. In Level 2, the agent identifies the task responsible for the error, applies a revision function from the solution library, updates the knowledge base, and compiles a detailed explanation for the user. All natural-language messages generated by the agent in the diagram, including the step-by-step explanations, are produced by ChatGPT (individual arrows for ChatGPT calls are omitted for clarity).

Algorithm 1: Task Localization

Input: Feedback F, TMK M, knowledge base K_b, user identifier U

Output: Task T, user data U_{data}, entities E_f, reasoning R

```
1: Extract $E_f$ from $F$.
2: Compute similarity between $F$ and task descriptions in $M$.
3: Let $T \leftarrow$ task with highest similarity, with confidence $C$.
4: if $C < \tau$ then
5:     return "No relevant task", $C$.
6: end if
7: Retrieve $U_{data}$ from $K_b[U]$.
8: if $U_{data}$ is empty then
9:     return "No user data".
10: end if
11: Apply reasoning with $E_f$ and $U_{data}$ to determine the revision reason.
12: Append the revision reason to $R$.
13: return $(T, U_{data}, E_f, R)$.
```

Algorithm 2: Knowledge Revision

Input: Task T, user data U_{data}, entities E_f, reasoning R, solution library L, knowledge base K_b

Output: Revision message M

```
1: Receive $(T, U_{data}, E_f, R)$ from Algorithm 1.
2: Look up $Fn \leftarrow L[T]$.
3: if $Fn$ is empty then
4:     return "No function available".
5: end if
6: Apply $Fn(E_f, U_{data}, K_b)$ to update $K_b[U]$.
7: Append the revision result to $R$.
8: Compile $R$ into $M$.
9: return $M$.
```

edge base, including the stored entities and the user's original post. Finally, a reasoning module built with LangChain and ChatGPT analyzes feedback and user data to infer why a revision is needed and classify it into one of the revision-need types summarized in Table 1.

Knowledge Revision The agent revises its knowledge base using Algorithm 2. Using the localized task from Task Localization (Algorithm 1), the agent performs a dictionary lookup in a solution library to retrieve the associated revision function. We constructed this solution library from the revision-need types in Table 1 and their corresponding revision functions. The agent then applies the retrieved function to update the user data stored in the knowledge base.

During Task Localization and Knowledge Revision, each step produces an intermediate natural-language message that describes the agent's actions and reasoning, generated with LangChain and ChatGPT. At the end of Level 2, these messages are compiled into a single step-by-step revision message, which is delivered to the user (Figure 2).

Illustrative Scenario

Figure 2 illustrates a self-adaptation scenario in SAMI, demonstrating how the self-adaptation architecture operates in practice. During deployments in Georgia Tech's OMSCS program, SAMI has frequently misinterpreted users' prior locations as their current locations and generated social recommendations based on this misinterpretation. In the bottom dialogue (Level 1), the agent misinterprets "Seoul" from the user's post as the current location and recommends users in Seoul. In the top dialogue (Level 2), the user corrects the agent, which triggers self-adaptation. The agent extracts entities from the feedback, localizes the entity-extraction task as the source of the error, infers the revision reason, and updates its knowledge base. The agent then communicates this process back to the user through a step-by-step explanation.

Evaluation

Design

To demonstrate the architecture's feasibility across diverse scenarios before large-scale deployment, we employed a controlled synthetic data check (Nauta et al. 2023). We constructed 20 cases based on real student data and revision-need types observed during deployment (Table 1). We assigned five instances to each of the four open revision-need types: Hallucination, Omission, Misinterpretation, and User-Initiated Update.

We generated these cases using the following three-step process: (1) extracting real student data from the knowledge base, (2) conditionally injecting an error based on the revision-need type, and (3) creating minimal user feedback. For step (2), no error was added if the revision-need type was a User-Initiated Update. For step (3), we provided brief feedback to reflect how users typically respond in real deployments. Each case included the knowledge base and the corresponding user feedback.

We evaluated the architecture along three dimensions: Localization, Revision, and Explanation. Downstream stages were assessed independently, so Revision or Explanation could receive a point even if Localization failed. A case was considered a complete success only if it achieved all three points, reflecting end-to-end effectiveness.

- **Localization (1 point)**: Did the agent correctly identify the task and error type?
- **Revision (1 point)**: Did the agent successfully revise the knowledge base as intended?
- **Explanation (1 point)**: Did the agent generate a correct and complete explanation of the revision?

For Explanation, correctness was judged as "nothing but the truth" (yes/partial/no), and completeness as "the whole truth" (complete/incomplete) (Nauta et al. 2023). All explanations were manually evaluated to ensure a fine-grained and reliable assessment.

For example, in a Hallucination case, we injected "New York" as the user's primary location into the knowledge base, even though it did not appear in the post. The corresponding user feedback was, "*I don't live in New York.*" This case received one point for Localization when the agent

Type	**Locali-zation**	**Revision**	**Explan-ation**	**Complete Success**
Hallucination	0.6	1.0	1.0	0.6
Omission	0.6	0.8	1.0	0.6
Misinterpretation	0.8	0.8	1.0	0.8
User Update	1.0	1.0	1.0	1.0
Average	0.75	0.9	1.0	0.75

Table 3: Evaluation scores across observed revision-need types with 20 feedback cases. 15 of the 20 achieved all three evaluation points. The performance illustrates the architectural benefits of integrating KBAI and LLMs. Lower localization scores reflect the use of minimal and informal user feedback to approximate real deployment conditions.

correctly identified the task as Identify and Extract Entities from Ed Post, and the error type as Hallucination. It received one point for Revision when the agent removed "New York" from its knowledge base, and one point for Explanation when the agent generated a complete revision message without omissions or errors.

Results

The self-adaptation architecture was effective in our evaluation, with 15 out of 20 cases achieving complete success (i.e., all three evaluation points). Table 3 summarizes the scores across revision-need types.

Performance was strong for Revision (0.9 average) and Explanation (1.0), indicating that the architecture reliably corrects errors and clearly communicates revisions. Localization scores were lower (0.75), mainly because the evaluation intentionally used brief feedback to reflect authentic user behavior (e.g., "*It's ML for Trading*"). This design limited the contextual cues available to the agent. In a preliminary analysis, when we provided more detailed feedback, the architecture achieved perfect performance across all 20 cases (e.g., "*It's Machine Learning for Trading, not just Machine Learning*").

These results highlight several advantages of combining KBAI and LLMs. TMK structures the agent's internal process into smaller tasks, making it easier for LLMs to pinpoint where revisions are needed. This decomposition guides the agent toward more consistent outputs (Goel and Rugaber 2015; Murdock and Goel 2008). Similarly, the solution library and knowledge graph provide explicit symbolic structure that grounds the LLMs' outputs. This grounding enables more reliable and precise agent behavior (Pan et al. 2024; Gaur and Sheth 2024).

The agent showed suboptimal performance when feedback included multiple entities, frequently extracting only one (e.g., "*I'm not in New York anymore, and I never mentioned hiking*"). Although the agent rarely makes more than one mistake at a time, users may issue multiple updates simultaneously. Hence, future work will extend the architecture to support more complex feedback types and multi-entity corrections.

Path to Deployment

The self-adaptation architecture is scheduled for deployment in Spring 2026 in Georgia Tech's OMSCS program, one of the world's largest online graduate programs. Over the past ten semesters, SAMI has been deployed at scale, and its TMK-based self-explanation feature is already in active use (Basappa et al. 2024). Building on this deployed foundation, we will integrate the self-adaptation architecture into the Knowledge-Based AI course, which enrolls over 500 students each semester. We anticipate improvements in recommendation accuracy and user satisfaction by involving users directly in correcting and shaping the knowledge base.

This deployment will allow us to assess how effectively the architecture diagnoses and corrects errors in real-world use, beyond the controlled evaluation (Table 3). It will also allow us to examine how metacognitive adaptation influences trust and long-term engagement. In doing so, the deployment extends the theoretical insights from the foundational perception study (Wang 2024) into an in-situ context. We will use a mixed-methods design to capture both agent effectiveness and users' real-world experiences.

Conclusion

LLM-based AI agents inevitably make mistakes, and such errors can undermine user trust and discourage continued use. In our real-world deployments of SAMI in Georgia Tech's OMSCS program, we observed that the agent occasionally produces incorrect outputs due to ChatGPT's fallibility. Users also frequently request updates as their circumstances change (Table 1). In both cases, the agent must adapt to user feedback and communicate revisions transparently to maintain robustness and trust.

Our two-level metacognitive self-adaptation architecture addresses this need (Figure 1). It localizes LLM-induced errors, revises the underlying knowledge base, and communicates the revision process through step-by-step explanations. By integrating TMK and a knowledge graph with LLM-based reasoning, the architecture combines symbolic structure with generative flexibility to support reliable and interpretable adaptation (Figure 2). Evaluation on 20 feedback cases drawn from real student data demonstrates its feasibility across diverse scenarios (Table 3). Looking ahead, the upcoming deployment will assess its long-term effectiveness in real-world use.

Beyond SAMI, this work offers a generalizable architecture for developing robust and trustworthy AI agents. It shows how a metacognitive layer can support introspection over an agent's reasoning process, and how KBAI and LLMs can complement each other to enable robust self-adaptation. Our work opens new directions for developing AI agents that continually improve themselves while keeping users informed about how and why they adapt.

Acknowledgments

We thank Spencer Rugaber and Rhea Basappa for their contributions to TMK and self-explanation in SAMI, respectively. This research was supported by a grant from the US

NSF (#2247790) to the National AI Institute of AI for Adult Learning and Online Education (aialoe.org).

References

Ashktorab, Z.; Jain, M.; Liao, Q. V.; and Weisz, J. D. 2019. Resilient Chatbots: Repair Strategy Preferences for Conversational Breakdowns. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 1–12. New York, NY, USA: Association for Computing Machinery.

Bang, Y.; Cahyawijaya, S.; Lee, N.; Dai, W.; Su, D.; Wilie, B.; Lovenia, H.; Ji, Z.; Yu, T.; Chung, W.; Do, Q. V.; Xu, Y.; and Fung, P. 2023. A Multitask, Multilingual, Multimodal Evaluation of ChatGPT on Reasoning, Hallucination, and Interactivity. arXiv:2302.04023.

Basappa, R.; Tekman, M.; Lu, H.; Faught, B.; Kakar, S.; and Goel, A. K. 2024. Social AI Agents Too Need to Explain Themselves. In *Generative Intelligence and Intelligent Tutoring Systems*, 351–360. Cham: Springer Nature Switzerland.

Cox, M.; and Raja, A. 2007. Metareasoning: A manifesto. *BBN Technical*.

Cox, M. T. 2005. Metacognition in computation: A selected research review. *Artificial Intelligence*, 169(2): 104–141.

Erol, K.; Hendler, J.; and Nau, D. S. 1994. HTN planning: Complexity and expressivity. In *AAAI*, 1123–1128.

Ganapini, M. B.; Campbell, M.; Fabiano, F.; Horesh, L.; Lenchner, J.; Loreggia, A.; Mattei, N.; Rossi, F.; Srivastava, B.; and Venable, K. B. 2023. Thinking Fast and Slow in AI: The Role of Metacognition. In *Machine Learning, Optimization, and Data Science*, 502–509. Cham: Springer Nature Switzerland.

Gaur, M.; and Sheth, A. 2024. Building trustworthy NeuroSymbolic AI Systems: Consistency, reliability, explainability, and safety. *AI Magazine*, 45(1): 139–155.

Goel, A. K.; and Rugaber, S. 2015. Interactive Meta-Reasoning: Towards a CAD-Like Environment for Designing Game-Playing Agents. In Besold, T. R.; Schorlemmer, M.; and Smaill, A., eds., *Computational Creativity Research: Towards Creative Machines*, 347–370. Paris: Atlantis Press. ISBN 978-94-6239-085-0.

Goel, A. K.; and Rugaber, S. 2017. GAIA: A CAD-Like Environment for Designing Game-Playing Agents. *IEEE Intelligent Systems*, 32(3): 60–67.

Hoang, H.; Lee-Urban, S.; and Muñoz Avila, H. 2005. Hierarchical plan representations for encoding strategic game AI. In *Proceedings of the First AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 63–68. AAAI Press.

Honig, S.; and Oron-Gilad, T. 2018. Understanding and Resolving Failures in Human-Robot Interaction: Literature Review and Model Development. *Frontiers in Psychology*, 9.

Ji, Z.; Lee, N.; Frieske, R.; Yu, T.; Su, D.; Xu, Y.; Ishii, E.; Bang, Y. J.; Madotto, A.; and Fung, P. 2023. Survey of Hallucination in Natural Language Generation. *ACM Comput. Surv.*, 55(12).

Kakar, S.; Basappa, R.; Camacho, I.; Griswold, C.; Houk, A.; Leung, C.; Tekman, M.; Westervelt, P.; Wang, Q.; and Goel, A. K. 2024. SAMI: An AI Actor for Fostering Social Interactions in Online Classrooms. In *Generative Intelligence and Intelligent Tutoring Systems*, 149–161. Cham: Springer Nature Switzerland.

Kirk, J. R.; Wray, R. E.; Lindes, P.; and Laird, J. E. 2024. Improving Knowledge Extraction from LLMs for Task Learning through Agent Analysis. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16): 18390–18398.

Lawley, L.; and Maclellan, C. 2024. VAL: Interactive Task Learning with GPT Dialog Parsing. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery.

Lee, Y.; Son, K.; Kim, T. S.; Kim, J.; Chung, J. J. Y.; Adar, E.; and Kim, J. 2024. One vs. Many: Comprehending Accurate Information from Multiple Erroneous and Inconsistent AI Generations. In *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*, 2518–2531. New York, NY, USA: Association for Computing Machinery.

Lee-Urban, S.; and Muñoz-Avila, H. 2006. A Study of Process Languages for Planning Tasks. *ICAPS 2006*.

Murdock, J. W.; and Goel, A. K. 2008. Meta-case-based reasoning: self-improvement through self-understanding. *Journal of Experimental & Theoretical Artificial Intelligence*, 20(1): 1–36.

Nau, D. S.; Au, T.-C.; Ilghami, O.; Kuter, U.; Murdock, J. W.; Wu, D.; and Yaman, F. 2003. SHOP2: An HTN planning system. *Journal of artificial intelligence research*, 20: 379–404.

Nauta, M.; Trienes, J.; Pathak, S.; Nguyen, E.; Peters, M.; Schmitt, Y.; Schlötterer, J.; van Keulen, M.; and Seifert, C. 2023. From Anecdotal Evidence to Quantitative Evaluation Methods: A Systematic Review on Evaluating Explainable AI. *ACM Comput. Surv.*, 55(13s).

Pan, S.; Luo, L.; Wang, Y.; Chen, C.; Wang, J.; and Wu, X. 2024. Unifying Large Language Models and Knowledge Graphs: A Roadmap. *IEEE Transactions on Knowledge and Data Engineering*, 36(7): 3580–3599.

Salem, M.; Lakatos, G.; Amirabdollahian, F.; and Dautenhahn, K. 2015. Would You Trust a (Faulty) Robot? Effects of Error, Task Type and Personality on Human-Robot Cooperation and Trust. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, 141–148. New York, NY, USA: Association for Computing Machinery.

Schmill, M.; Oates, T.; Anderson, M. L.; Josyula, D.; Perlis, D.; Wilson, S.; and Fults, S. 2008. The role of metacognition in robust AI systems. In *Workshop on Metareasoning at the Twenty-Third AAAI Conference on Artificial Intelligence*.

Wang, Q. 2024. *Mutual theory of mind for human-AI communication in AI-mediated social interaction*. Ph.d. dissertation, Georgia Institute of Technology.

Wang, Q.; Jing, S.; and Goel, A. K. 2022. Co-Designing AI Agents to Support Social Connectedness Among Online

Learners: Functionalities, Social Characteristics, and Ethical Challenges. In *Proceedings of the 2022 ACM Designing Interactive Systems Conference*, 541–556. New York, NY, USA: Association for Computing Machinery.