

000 RAG KNOWLEDGE ONLINE CORRECTION WITH 001 CONVERSATION-BASED USER FEEDBACK 002 003 004

005 **Anonymous authors**

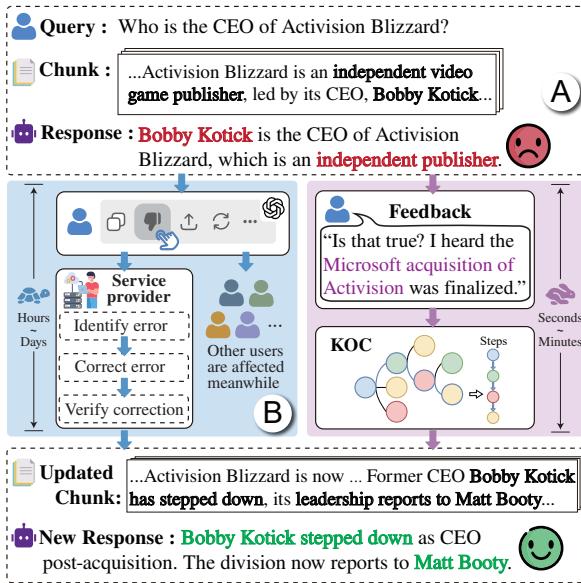
006 Paper under double-blind review

007 ABSTRACT

011 Retrieval-Augmented Generation (RAG) is a promising way to enhance LLMs
012 by integrating external knowledge. However, its performance degrades when the
013 knowledge contains errors. When users encounter such errors, the typical cor-
014 rection process involves users reporting the errors, after which server providers
015 investigate the knowledge base to identify and fix the issues. This process is often
016 time-consuming, and in the meantime, other users continue to encounter the same
017 errors, leading to poor user experience. To address this challenge, we propose
018 a new task, **Knowledge Online Correction**, which focuses on correcting errors
019 immediately after they are pointed out by users through conversation-based feed-
020 back. To evaluate this task, we conducted a preliminary user study and developed
021 a new benchmark, **ConvCorrect**. To address this task, we propose a Multi-step
022 Knowledge Online Correction method(**MT-KOC**), an online knowledge correction
023 method that automatically corrects errors in real time based on a dynamic action
024 search algorithm. Empirical results shows that MT-KOC outperforms baseline
025 methods, achieving higher accuracy in the knowledge online correction task.

026 1 INTRODUCTION

029 Large Language Models (LLMs) have
030 achieved remarkable success in real world
031 applications, but still face challenges
032 such as hallucination and outdated knowl-
033 edge (Huang et al., 2025; Zhang et al.,
034 2023). Retrieval-Augmented Generation
035 (RAG) has emerged as a promising solution
036 by integrating external knowledge to miti-
037 gate these issues (Lewis et al., 2020; Gao
038 et al., 2023b). However, its performance
039 degrades when the underlying knowledge
040 base contains errors (e.g., Fig. 1A). When
041 users encounter such errors, the typical
042 correction process involves tedious man-
043 ual intervention: users report the issue,
044 and server providers investigate the knowl-
045 edge base to identify and correct the
046 errors (Fig. 1B). This process is often time-
047 consuming, potentially taking several hours
048 to multiple working days (McGraw, 2025).
049 During this period, the knowledge errors
050 persist and continue to mislead users, re-
051 sulting in a poor overall experience. To
052 address the issue of inefficiency, some re-
053 cent studies have proposed to automatically correct the errors in real time through internet retrieval
(e.g., Wikipedia) (Yan et al., 2024). However, the effectiveness of these methods depends on the
accuracy of the internet retrieval results. If the internet retrieval results are incorrect, these methods
have no alternative mechanism to resolve the errors.



054 Figure 1: Comparison between slow **manual correction**
055 (left) and rapid **knowledge online correction** (right).

To tackle this issue, we drew inspiration from the **Formative Assessment with Feedback** model in education (McKenzie et al., 2017). In this model, teachers provide real-time, conversation-based feedback during discussions, enabling students to quickly identify and correct misconceptions and thereby accelerate learning. Motivated by this model, we propose a new strategy, where users act as teachers, offering conversation-based feedback when they identify errors. For example, users can indicate whether an error exists and suggest how it should be corrected. Such feedback can serve as additional guidance for error correction, helping to overcome the limitations of existing methods that rely solely on internet retrieval results. Based on this new strategy, we define a new task, Knowledge Online Correction (KOC), which leverages conversation-based user feedback to correct knowledge errors in real time.

While this new strategy has the potential to help correct knowledge errors in real time, how to effectively leverage user feedback is still unclear, presenting two major challenges. **Challenge 1:** There are no established benchmarks for this task. History has demonstrated that high-quality benchmarks (e.g., ImageNet (Deng et al., 2009), GLUE (Wang et al., 2019)) can effectively advance the development of models for specific tasks. However, developing benchmarks for the KOC task is non-trivial due to the diverse and open-ended nature of conversation-based feedback. Therefore, how to ensure that the benchmark is both comprehensive and representative remains an open question. **Challenge 2:** It is unclear how to effectively leverage the conversation-based user feedback for error correction. Conversation-based feedback can take many forms, from precise, well-structured corrections to vague hints that an error exists. Effectively distinguishing useful feedback from noisy or misleading input is critical, as is determining how to integrate this feedback into model updates in a way that effectively improves performance.

In this paper, we first developed a new benchmark, ConvCorrect (**Challenge 1**). To construct this benchmark, we first conducted a preliminary user study. From this preliminary study, we summarized five error types in the knowledge bases and how users provide feedback for these errors. Based on this summarization, we synthesize errors in the knowledge bases and the conversation-based feedback, which was further verified by humans to ensure quality. Based on this benchmark, we developed MT-KOC for the KOC task (**Challenge 2**). MT-KOC uses a dynamic action search algorithm to identify an optimal sequence of edits for progressive correction, ensuring better alignment with conversation-based feedback. The experiments show that our method outperforms the existing SOTA methods by 5.88% on average in terms of F1-score.

In summary, our contributions are threefold:

1. We develop ConvCorrect, the first benchmark designed and supported by a preliminary user study to evaluate knowledge online correction methods.
2. We propose MT-KOC, a method based on a dynamic action search algorithm to address the challenges of online knowledge base correction.
3. We validate our approach through empirical experiments, demonstrating the effectiveness of MT-KOC, and through a user study that confirms the rational design of our benchmark.

2 RELATED WORK

2.1 RETRIEVAL-AUGMENTED GENERATION

Retrieval-Augmented Generation (RAG) has been established as an effective paradigm for mitigating hallucinations in Large Language Models (LLMs) (Brown et al., 2020; Chowdhery et al., 2022; Touvron et al., 2023) by grounding them in external knowledge (Lewis et al., 2020; Gao et al., 2023b; Guu et al., 2020; Izacard & Grave, 2021). Extensive research has focused on optimizing the RAG pipeline, with significant advancements in query formulation (Zheng et al., 2024; Ma et al., 2023), dynamic and adaptive retrieval strategies (Asai et al., 2024; Jeong et al., 2024), and post-retrieval context refinement (Yu et al., 2024; Li et al., 2025; Xu et al., 2024).

However, a majority of these methods operate under the assumption that the knowledge source itself is accurate. Their objective is to better *utilize* existing knowledge, not to *correct* the knowledge base. While a parallel line of research explores editing the model’s internal, parametric knowledge (Meng et al., 2023; Zhang et al., 2024), this is distinct from correcting external, non-parametric knowledge

108 sources. Advanced frameworks (Yan et al., 2024; Wang et al., 2025) focus on rectifying the answer
 109 for a single turn, rather than propagating the fix back to the underlying source. To address this critical
 110 gap, we introduce the novel task of **Knowledge Online Correction (KOC)**, which aims to rectify
 111 the knowledge source in real-time, driven by user feedback.
 112

113 2.2 MULTI-AGENT SYSTEMS

114
 115 Multi-Agent Systems (MAS), wherein complex tasks are decomposed and solved by multiple
 116 collaborative agents, have demonstrated significant efficacy and have been widely explored (Wang
 117 et al., 2024; Xi et al., 2025). Recent advancements in MAS have progressed along two main
 118 fronts: developing increasingly sophisticated collaborative frameworks, evolving from structured
 119 workflows to large-scale engineering platforms (Hong et al., 2024; Qian et al., 2024), and enhancing
 120 the individual agent’s capabilities with mechanisms for complex reasoning, reflection, and self-
 121 correction (Yao et al., 2023b; Shinn et al., 2023; Gou et al., 2024; Madaan et al., 2023).
 122

123 Our approach is inspired by the core MAS principle of solving problems via a sequence of planned
 124 actions. However, our work introduces key innovations. Most MAS are **generative**, tasked with
 125 creating content like code or reports. While some agents exhibit self-correction, this is typically
 126 aimed at refining their own generated output. We, in contrast, use this paradigm for a corrective
 127 task—our KOC, which targets an external knowledge source. Furthermore, our system is designed to
 128 be **feedback-driven**, specifically for the purpose of dynamic knowledge correction in RAG systems.
 129

130 3 BENCHMARK

131 3.1 PROBLEM FORMULATION

132 The KOC task is formally defined as follows. Given a user query Q , a knowledge chunk K is
 133 retrieved first, which includes errors. Based on the chunk K , an answer A is generated by an LLM:
 134 $A = \text{LLM}(Q, K)$. After receiving the answer, the user responds with conversation-based feedback
 135 F . Then, the goal of the KOC task is to find a transformation function T such that the answer
 136 generated based on the correct version K_{oracle} is similar to the answer generated based on the chunk
 137 transformed by T :

$$138 \quad \begin{aligned} T^* = \operatorname{argmin}_T \quad & \text{Diff}\{\text{LLM}(Q, K^*), \text{LLM}(Q, K_{\text{oracle}})\} \\ 139 \quad \text{s.t.} \quad & K^* = T(K, F) \end{aligned} \quad (1)$$

140 Diff $\{\cdot, \cdot\}$ measures the difference between two answers generated by the LLM.
 141

142 3.2 PRELIMINARY USER STUDY

143 A key challenge in constructing the benchmarks for the KOC task lies in ensuring their comprehensiveness
 144 and representativeness. To achieve this, it is essential to understand how users typically provide
 145 conversation-based feedback when they encounter errors in LLM-generated answers. Therefore, we
 146 conducted a preliminary user study. Based on this study, we identified five common types of errors in
 147 LLM answers and further summarized typical conversation-based feedback corresponding to each
 148 error type.
 149

150 **Study setup.** We recruited 12 graduate students majoring in Computer Science for the user study,
 151 aged from 21 to 29 years (mean = 24.25, SD = 2.60). All of them have more than one year of
 152 experience in developing or using RAG systems. Upon completion, each participant received a \$10
 153 compensation, independent of their performance.

154 **Datasets.** The study was constructed on 1,200 samples randomly selected from the Neural-Bridge
 155 Dataset (Neural Bridge AI, 2024b;a). Each sample contains a query Q and a correct knowledge
 156 chunk K_{oracle} . Then, the chunk of each sample was perturbed to contain errors. To ensure the
 157 errors encompass most real-world scenarios, the perturbations are capable of generating any type
 158 of error. In the field of databases, updating, adding, and removing are used to cover all types of
 159 modifications. Motivated by this, we also include three types of perturbations Silberschatz et al.
 160 (2002). (1) Update: A key entity (e.g., date, name, location) in the relevant section is updated to
 161

162 an incorrect but plausible entity of the same type. (2) *Add*: A contextually coherent but factually
 163 incorrect or misleading sentence is added after the sentence containing the relevant information.
 164 (3) *Delete*: A critical phrase or sentence required to answer the query is removed from the relevant
 165 section. The chunk of each sample was randomly applied one of the three types of perturbations,
 166 resulting in an erroneous knowledge chunk (K). Based on K , an answer (A) was generated by an
 167 LLM.

168 **Procedure.** The 1,200 samples were randomly distributed to the 12 participants, with each participant
 169 receiving 100 samples. For each sample, the participants were asked to assume the role of a user who
 170 had received an erroneous answer (A) from a conversational AI system after posing a query. Their
 171 task was to write a single conversation-based feedback message (F) in response to the erroneous
 172 answer, reflecting what they would realistically provide in such a scenario.

173 **Error type.** After the user study, two researchers from our team were tasked with classifying the
 174 error type of the answer (A) in each of the 1200 samples. Each annotator was responsible for half
 175 of the data, and a cross-review was conducted upon completion to ensure consistency. This process
 176 resulted in the classification of all samples into five primary error types: **Fully Incorrect, Partially**
 177 **Incorrect, Fully Missing, Partially Missing, and Mixed**. For illustrative examples of each error
 178 type, please refer to Appendix A.

180 **Typical conversation-based feedback styles.** After the user study was completed, the same two
 181 expert annotators analyzed the feedback (F) provided by the participants. Following a similar process
 182 of independent annotation and cross-review, they categorized the collected feedback into seven
 183 distinct styles. Our analysis revealed that different error types elicited different distributions of
 184 feedback styles, as summarized in Table 1.

186 Error type	187 Associated feedback styles
188 Fully incorrect	189 Direct correction, Error indication
190 Partially incorrect	191 Direct correction, Error indication, Error localization
192 Fully missing	193 Direct completion, Missing indication
Partially missing	Direct completion, Missing indication
Mixed	Direct correction and completion, Error and missing indication

194 Table 1: Mapping from Error Types to Feedback Styles derived from the user study.

196 A key finding is that the relationship between error types and feedback styles is a complex one-
 197 to-many mapping. This discovery highlights a fundamental challenge: a simple, low-level textual
 198 perturbation can manifest as a wide spectrum of high-level, user-perceived semantic errors. This
 199 non-trivial mapping proves that a simplistic approach—mechanically generating feedback based on
 200 the perturbation type—would be unrealistic and fail to capture the task’s true complexity. Therefore,
 201 this empirically derived taxonomy not only reveals the task’s difficulty but also serves as the principled
 202 blueprint for our benchmark construction.

204 3.3 BENCHMARK CONSTRUCTION

206 Based on the error types and typical conversation-based feedback styles identified in the preliminary
 207 user study, we developed the first benchmark for the KOC task, ConvCorrect.

208 **Data Source and Pre-processing.** We construct ConvCorrect upon two diverse RAG benchmarks:
 209 SQuAD (Rajpurkar et al., 2016) and Neural-Bridge (Neural Bridge AI, 2024b;a), resulting in two
 210 splits: ConvCorrect-SQ and ConvCorrect-NB. We first pre-processed these source datasets to create a
 211 clean base for perturbation. For SQuAD, which features multiple questions per context, we retained
 212 only one unique sample per context, merging the training and test sets to yield 20,958 initial samples.
 213 For Neural-Bridge, we combined its two versions (RAG Dataset 12000 & RAG Dataset 1200) and
 214 de-duplicated them to obtain 12,893 unique samples. Finally, both datasets underwent sensitive
 215 content filtering. This resulted in a final set of 19,903 pristine samples from SQuAD and 12,000 from
 Neural-Bridge, which formed the basis for our benchmark construction.

216 **Construction pipeline.** Based on the pre-processed datasets, we constructed ConvCorrect in two
 217 main steps: **error generation** and **feedback generation**.
 218

- 219 • **Error generation.** Since the knowledge chunks in the source datasets contain no errors, we
 220 first perturbed the chunks to contain errors. Similar to the preliminary user study, we consider
 221 three types of perturbations here. For each chunk, all three types of perturbations were
 222 applied using LLMs (Ding et al., 2024), resulting in three distinct perturbed versions. These
 223 perturbed chunks were then classified into the five error types identified in the preliminary
 224 user study, also using LLMs. Finally, both the perturbed chunks and their corresponding
 225 error classifications were verified by two human evaluators.
- 226 • **Feedback generation.** Finally, to comprehensively model the diversity of user responses,
 227 we exhaustively generate all applicable feedback styles for *each* classified error. An LLM
 228 acting as a **User Simulator** takes the error type label from Stage 2 and, based on the
 229 one-to-many mapping from our user study (Table 1), generates multiple feedback messages
 230 (F) in different styles. For instance, for an error classified as Partially incorrect, the system
 231 generates separate samples with Direct Correction, Error Indication, and Error Localization
 232 feedback styles.

233 **Statistics.** The three-stage simulation pipeline, particularly the exhaustive generation strategy in
 234 Stage 1 and Stage 3, culminates in the final ConvCorrect benchmark, comprising **124,531** samples
 235 for ConvCorrect-SQ and **80,772** for ConvCorrect-NB. This strategy ensures our benchmark has high
 236 coverage and diversity in its error-feedback pairings. The detailed statistics of the Benchmark’s scale
 237 and type distribution are presented in Appendix Table 5.

239 4 METHOD

241 4.1 FRAMEWORK OVERVIEW

243 To tackle the KOC task, we introduce the **MuLTi-step KOC** (MT-KOC) framework. The design
 244 of MT-KOC is inspired by the typical correction process followed by humans. Generally, humans
 245 first retrieve relevant information from reliable sources, then use this information to correct specific
 246 chunks, and finally verify the accuracy of the corrected chunks. Since errors may be diverse and
 247 cannot always be resolved in a single step in practice, this process may be repeated multiple times.
 248 Therefore, we design a multi-agent system within MT-KOC to address the KOC task through multiple
 249 steps. Leveraging this system, we propose a dynamic action search algorithm that identifies the
 250 optimal sequence of corrective actions to accurately correct the chunks.

252 4.2 MULTI-AGENT SYSTEM

253 The multi-agent system consists of four collaborative agents: a **Knowledge Distillation Agent**, an
 254 **Action Recommendation Agent**, a **Correction Agent**, and a **Reward Evaluation Agent**.
 255

256 **Knowledge Distillation Agent.** The correction of knowledge chunks relies on retrieval results,
 257 such as those from the Internet. Therefore, the agent retrieves relevant information for the chunks
 258 to be corrected. It takes a user query (Q) and the initial Knowledge Chunk ($K_0 = K$) as input to
 259 extract information (I). This process is represented by the function f_{distill} :

$$I = f_{\text{distill}}(Q, K_0) \quad (2)$$

262 The function f_{distill} is realized by the agent, which is an LLM guided by the prompt detailed in
 263 Appendix B.1. It is important to note that this agent is not limited to retrieving information from the
 264 Internet; it can also retrieve from LLMs, also known as knowledge distillation.

266 **Action Recommendation Agent.** As pointed out in a recent survey (Tran et al., 2025), the per-
 267 formance of LLMs would be improved if a complex action is decomposed into several sub-actions.
 268 Therefore, we also decompose the correction action into three commonly used sub-actions: ADD,
 269 DELETE, and REVISE, which are called actions. ADD: Complete missing information in Knowledge
 Chunk, where the agent specifies the exact information and insertion position. DELETE: Remove

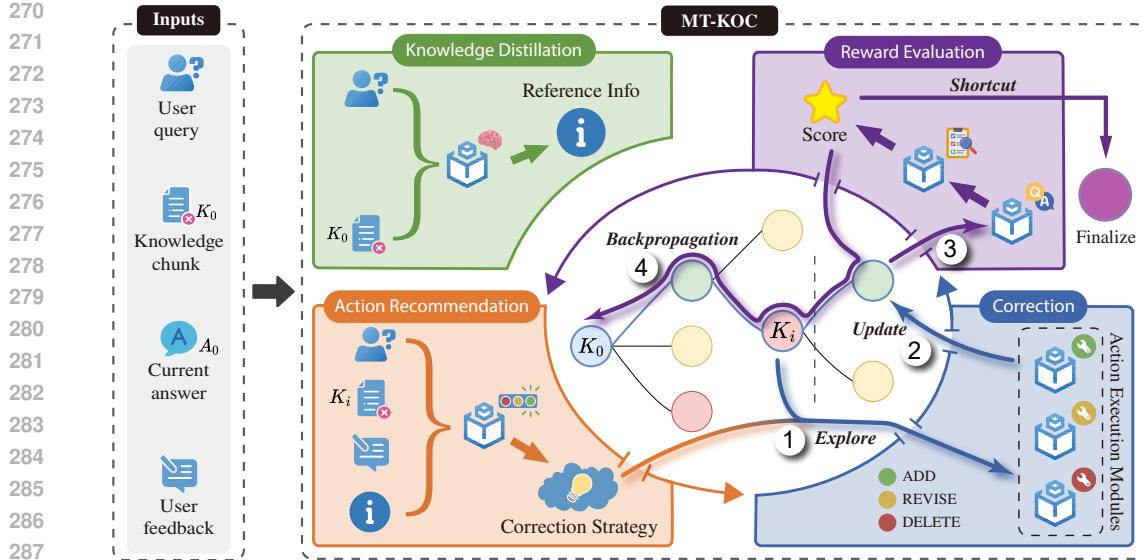


Figure 2: Overview of the MT-KOC framework, illustrating the collaboration across four key agents. The process iteratively explores a search space of possible edits to find an optimal correction path for the Knowledge Chunk, guided by user feedback and a reward-driven mechanism.

redundant or misleading information in the Knowledge Chunk that could interfere with generating the correct answer, with the agent explicitly identifying the target. **REVISE**: Modify inaccurate or poorly phrased information in the Knowledge Chunk, where the agent provides a revised version. Accordingly, the correction can be represented by a sequence of actions $\{C_1, \dots, C_k\}$. Therefore, action recommendation agents recommend suitable actions given the user query (Q), current knowledge chunk (K), user feedback (F), and reference information (I). The agent's recommendation logic is implemented using an LLM, its specific prompt is provided in Appendix B.2.

Correction Agent. This agent acts as the executor. It is a composite of three specialized modules corresponding to the action types (ADD, DELETE, REVISE). When a single correction action C_i is selected by the Recommendation Agent, this agent dispatches it to the appropriate module for execution, producing the next Knowledge Chunk state, K_{i+1} :

$$K_{i+1} = f_{\text{correct}}(K_i, C_i) \quad (3)$$

The function f_{correct} represents this execution, each module within this agent is also guided by an LLM prompt, as detailed in Appendix B.3.

Reward Evaluation Agent. This agent evaluate the corrections based on the conversation-based user feedback (F) and reference information (I) (Zheng et al., 2023). It takes the new Knowledge Chunk state K_{i+1} and generates the answer A_{i+1} by using LLMs. It then assesses this answer based on the query (Q), feedback (F), and reference information (I) to produce a numerical score, S_{i+1} :

$$S_{i+1} = f_{\text{eval}}(K_{i+1}, Q, F, I) \quad (4)$$

The evaluation function f_{eval} is implemented by the agent, which is prompted to act as an impartial judge. The full prompt, which specifies the scoring criteria and scale, can be found in Appendix B.4.

4.3 DYNAMIC ACTION SEARCH

Based on the multi-agent system, it is required to identify the optimal sequence of corrective actions to accurately correct the chunks. Therefore, the transformation T is realized by finding an optimal sequence of discrete correction actions, which we denote as \mathbf{C}^* . Let $\text{Apply}(K_0, \mathbf{C})$ be the function that returns the final Knowledge Chunk after applying the entire sequence \mathbf{C} to the initial chunk K_0 . The optimal sequence \mathbf{C}^* is the one that maximizes the expected reward of this final chunk:

$$\mathbf{C}^* = \arg \max_{\mathbf{C}} \mathbb{E} [f_{\text{eval}}(\text{Apply}(K_0, \mathbf{C}))] \quad (5)$$

324 **1. Initialization.** The process begins with the initial erroneous Knowledge Chunk, K_0 , which forms
 325 the root node of the search tree. The Knowledge Distillation Agent is invoked once to generate a
 326 global Reference Information (I). Subsequently, the Action Recommendation Agent generates the
 327 first set of candidate corrections from this root node.

328 **2. Iterative Search.** For a predefined number of Search Epochs, the framework executes a search
 329 loop consisting of the following steps:

331 *Exploration:* In each iteration, the Action Recommendation Agent recommends a set of candidate
 332 actions, $\mathcal{C}(K_i)$, based on the current Knowledge Chunk state, represented by the current node K_i
 333 in the search tree. An action is then selected from the children using the Upper Confidence Bound
 334 applied to Trees (UCT) policy (Kocsis & Szepesvári, 2006; Yao et al., 2023a):

$$335 \quad C_{i+1} = \arg \max_{C \in \mathcal{C}(K_i)} \left(\frac{R(C)}{N(C)} + c \sqrt{\frac{\ln N(K_i)}{N(C)}} \right) \quad (6)$$

338 Here, $\arg \max$ selects the action C_{i+1} from the candidate set $\mathcal{C}(K_i)$ that maximizes the UCT score.
 339

340 The UCT score balances exploitation and exploration: the term $\frac{R(C)}{N(C)}$ represents the average reward
 341 of action C , encouraging the selection of actions with historically high rewards, while $c \sqrt{\frac{\ln N(K_i)}{N(C)}}$
 342 promotes exploration of less-visited actions, where $R(C)$ is the total reward accumulated for action
 343 C , $N(C)$ is the number of times action C has been selected, $N(K_i)$ is the number of visits to the
 344 parent node K_i , and c is an exploration constant.

346 *Updating:* Once an action C_i is selected, the Correction Agent executes it to produce the next
 347 Knowledge Chunk state, K_{i+1} , which corresponds to a new node in the search tree.

348 *Evaluation:* The Reward Evaluation Agent then evaluates this new state to obtain a score S_{i+1} .

350 *Backpropagation:* This score is backpropagated up the search tree, updating the reward and visit
 351 counts for all actions along the path from the current node back to the root. Specifically, for each
 352 action C on the path: $R(C) \leftarrow R(C) + S_{i+1}$ and $N(C) \leftarrow N(C) + 1$.

354 **3. Final Selection.** The search process terminates, and an optimal corrected Knowledge Chunk (K^*)
 355 is selected based on one of two conditions:

356 *Shortcut Mechanism:* If at any point during the iterative search the Reward Evaluation Agent outputs
 357 a maximum score, the search is immediately terminated. The Knowledge Chunk corresponding to
 358 this high-reward path, represented by the corresponding node in the search tree, is directly returned
 359 as the optimal solution (K^*).

360 *Default Selection:* If the Shortcut is not triggered after all Search Epochs are completed, the system
 361 then determines the optimal correction sequence $\mathbf{C}^* = (C_0^*, C_1^*, \dots, C_L^*)$. This sequence is con-
 362 structed by starting from the root node ($K_0^* = K_0$) and iteratively selecting the correction with the
 363 highest average reward. At each step ‘ i ’, the selection of the next correction C_{i+1}^* is made from the
 364 children of the current optimal state K_i^* :

$$366 \quad C_{i+1}^* = \arg \max_{C \in \mathcal{C}(K_i^*)} \left(\frac{R(C)}{N(C)} \right) \quad (7)$$

368 Here, $\arg \max$ selects the action C_{i+1}^* with the highest average reward, ensuring a greedy selection
 369 of the best-performing actions to construct the optimal sequence. The final corrected Knowledge
 370 Chunk (K^*) is then obtained by applying this sequence \mathbf{C}^* to K_0 .

373 5 EXPERIMENTS

375 5.1 EXPERIMENTAL SETUP

377 **Datasets.** We evaluate our method on **ConvCorrect**, a benchmark we developed for interactive
 knowledge correction, consisting of two splits: ConvCorrect-SQ and ConvCorrect-NB (see Section 3

Method	ConvCorrect-SQ			ConvCorrect-NB		
	F1	Precision	Recall	F1	Precision	Recall
Perturbed	5.5	3.6	11.4	36.0	35.0	37.0
CRAG	23.1	13.8	71.6	52.0	50.3	53.8
RARR	23.0	13.8	68.3	52.2	50.4	54.2
Astute	23.5	14.0	72.8	53.0	50.2	56.2
MT-KOC	24.8	14.9	73.6	56.3	54.1	58.7
Oracle	30.3	18.2	89.8	64.7	61.3	68.6

Table 2: Aggregated F1-Scores, Precision, and Recall on the ConvCorrect-SQ and ConvCorrect-NB test sets. Comparison methods are framed by the performance bounds.

for details). Each sample includes the original knowledge chunk (K_{oracle}), a perturbed version (K_0), the query (Q), the erroneous answer (A_0), and user feedback (F).

Comparison Methods. To situate the performance of MT-KOC, we compare it against a diverse set of baselines, along with theoretical upper and lower bounds.

- **Lower Bound (Perturbed)** uses the initial, erroneous knowledge chunk (K_0) directly, establishing a performance floor.
- **Upper Bound (Oracle)** uses the ground-truth correct knowledge chunk (K_{oracle}), representing the theoretical performance ceiling.
- **CRAG** (Yan et al., 2024) employs a **self-correction** paradigm, using a lightweight retrieval evaluator to judge the necessity of refinement.
- **RARR** (Gao et al., 2023a) utilizes a **multi-step reasoning** approach, prompting the LLM to explicitly "research and revise" its outputs in a verifiable manner.
- **Astute** (Wang et al., 2025) tackles the problem via **knowledge conflict resolution**, identifying and resolving discrepancies between parametric and retrieved knowledge.

Evaluation Metrics. We evaluate performance from two perspectives:

- **Downstream Task Performance:** The primary metric, measuring the quality of the final answer. We compute token-level F1-Score, Precision, and Recall between the generated answer (A^*) and the oracle answer (A_{oracle}).
- **Knowledge Chunk Editing Quality:** A secondary metric assessing the precision of the edit itself. We use ROUGE-L (Y., 2004) to measure the similarity between the corrected chunk (K^*) and the original perturbed chunk (K_0), where a higher score indicates a more minimal, surgical correction.

Implementation Details. To ensure a fair comparison, all methods are powered by the same base model. We provide detailed hyperparameters and further implementation specifics in Appendix C.

5.2 RESULTS AND ANALYSIS

5.2.1 MAIN PERFORMANCE COMPARISON

The aggregated results on both test sets are reported in Table 2. The data clearly indicate that MT-KOC achieves the best performance across all downstream task metrics on both Benchmarks. The consistent performance hierarchy validates the superiority of our multi-step, adaptive search strategy over other structured refinement methods, significantly closing the gap to the Oracle performance.

Analysis of the Gap to Oracle. Despite MT-KOC's strong performance, a discernible gap to the Oracle upper bound persists. Our analysis attributes this gap to two primary challenges: 1) the inherent ambiguity of **vague user feedback** (F), which could be mitigated by multi-turn clarification dialogues; 2) scenarios where Knowledge Chunks (K_0) contain **specialized or lesser-known information**. Both cases compel the framework to fall back upon the base model's own parametric knowledge (Lewis et al., 2020). This indicates that the primary performance bottleneck lies not with the MT-KOC

Method	ConvCorrect-SQ		ConvCorrect-NB	
	v.s. Perturbed	v.s. Oracle	v.s. Perturbed	v.s. Oracle
CRAG	62.1	59.1	56.0	44.9
RARR	60.1	59.6	47.8	39.8
Astute	59.2	59.2	54.2	46.0
MT-KOC	79.7	78.4	63.1	53.4

Table 3: ROUGE-L similarity scores on the ConvCorrect-SQ and ConvCorrect-NB test sets.

framework itself, but with the knowledge breadth of the underlying LLM, a limitation that could be addressed by integrating real-time web search capabilities.

5.2.2 KNOWLEDGE CHUNK EDITING RESULTS ANALYSIS

To understand how each method edits the Knowledge Chunk, we analyze the ROUGE-L scores, shown in Table 3. Firstly, MT-KOC achieves the highest ROUGE-L score against the Oracle knowledge chunk. This directly measures the **accuracy** of the correction, demonstrating that the final edited chunk (K^*) is semantically closest to the ground-truth correct version. Secondly, and equally important, it also achieves the highest score against the original Perturbed chunk. This metric reflects **edit retention**, indicating that our method preserves the maximum amount of correct, unchanged information from the original chunk (K_0).

5.3 ABLATION STUDY

We conducted an ablation study to validate the contributions of MT-KOC’s key components, with results shown in Table 4.

Ablation on Multi-Path Search.

Removing multi-path exploration by degenerating the process into a linear, greedy search leads to a consistent degradation in performance. This confirms that the search mechanism is crucial for escaping local optima and finding a globally superior correction path.

Ablation on Shortcut Mechanism.

Disabling the Shortcut mechanism also causes a clear drop in F1-score, which suggests that the model becomes prone to performing superfluous, harmful edits without an effective early termination signal.

Method	Metric	SQ	NB
MT-KOC	F1	24.8	56.3
	P	14.9	54.1
	R	73.6	58.7
w/o M-P Search	F1	24.6 ($\downarrow 0.2$)	55.9 ($\downarrow 0.4$)
	P	14.8 ($\downarrow 0.1$)	53.6 ($\downarrow 0.5$)
	R	72.6 ($\downarrow 1.0$)	58.3 ($\downarrow 0.4$)
w/o Shortcut	F1	23.3 ($\downarrow 1.5$)	55.5 ($\downarrow 0.8$)
	P	14.0 ($\downarrow 0.9$)	53.2 ($\downarrow 0.9$)
	R	69.3 ($\downarrow 4.3$)	57.9 ($\downarrow 0.8$)

Table 4: Ablation study of MT-KOC. SQ and NB refer to the ConvCorrect-SQ and ConvCorrect-NB datasets, respectively. P/R denotes Precision/Recall.

6 CONCLUSION

In this paper, we introduced the task of KOC to correct errors in RAG knowledge bases in real-time using conversation-based user feedback. To address this task, we developed ConvCorrect, the first comprehensive benchmark that incorporates diverse error types and feedback styles derived from a preliminary user study. Furthermore, we proposed MT-KOC, a multi-step correction framework leveraging a multi-agent system and dynamic action search algorithm to identify and apply optimal sequences of corrective actions. Empirical evaluations on ConvCorrect demonstrate that MT-KOC significantly outperforms existing baselines, achieving higher correction accuracy.

486 **Ethics Statement** Our research includes a preliminary user study involving human participants. All
 487 participants were informed of the study’s objectives and procedures before providing their consent to
 488 participate. The data collected, which consists of conversation-based feedback, was fully anonymized
 489 to protect the privacy of the participants. Each participant received fair compensation for their time,
 490 independent of their performance. The datasets used as the foundation for our benchmark, SQuAD
 491 and Neural-Bridge, are established public resources, and we have taken additional steps to filter them
 492 for any potentially sensitive content.

493 **Reproducibility Statement** To ensure the reproducibility of our results, we provide detailed
 494 descriptions of our benchmark, methodology, and experimental setup. The construction process
 495 of our ConvCorrect benchmark, including data sources and processing pipelines, is detailed in
 496 Section 3. The architecture and the dynamic action search algorithm of our proposed MT-KOC
 497 method are described in Section 4. Full implementation details, including the base model used and
 498 hyperparameters, are provided in Appendix C. The source code and benchmark data will be made
 499 publicly available upon publication of this work.

500 **REFERENCES**

501 Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to
 502 retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on
 503 Learning Representations*, 2024.

504 Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal,
 505 Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel
 506 Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler,
 507 Jeffrey Wu, Clemens Winter, Chris Hesse, et al. Language models are few-shot learners. In
 508 *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901, 2020.

509 Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam
 510 Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker
 511 Schuh, Kensen Shi, et al. Palm: Scaling language modeling with pathways. *arXiv preprint
 512 arXiv:2204.02311*, 2022.

513 Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hier-
 514 archical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*,
 515 pp. 248–255, 2009.

516 Bosheng Ding, Chengwei Qin, Ruochen Zhao, Tianze Luo, Xinze Li, Guizhen Chen, Wenhan Xia,
 517 Junjie Hu, Anh Tuan Luu, and Shafiq Joty. Data augmentation using large language models: Data
 518 perspectives, learning paradigms and challenges. *arXiv preprint arXiv:2403.02990*, 2024.

519 Luyu Gao, Zhuyun Dai, Panupong Pasupat, Anthony Chen, Arun Tejasvi Chaganty, Yicheng Fan,
 520 Vincent Y. Zhao, Ni Lao, Hongrae Lee, Da-Cheng Juan, and Kelvin Guu. RARR: researching
 521 and revising what language models say, using language models. In *Proceedings of the 61st
 522 Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp.
 523 16477–16508. Association for Computational Linguistics, 2023a.

524 Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng
 525 Wang, and Haofen Han. Retrieval-augmented generation for large language models: A survey.
 526 *arXiv preprint arXiv:2312.10997*, 2023b.

527 Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen.
 528 CRITIC: large language models can self-correct with tool-interactive critiquing. In *The Twelfth
 529 International Conference on Learning Representations*, 2024.

530 Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. Realm: Retrieval-
 531 augmented language model pre-training. *arXiv preprint arXiv:2002.08909*, 2020.

532 Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao
 533 Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng
 534 Xiao, Chenglin Wu, and Jürgen Schmidhuber. Metagpt: Meta programming for A multi-agent

540 collaborative framework. In *The Twelfth International Conference on Learning Representations*,
 541 2024.

542

543 Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong
 544 Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A survey on hallucination in large
 545 language models: Principles, taxonomy, challenges, and open questions. *ACM Trans. Inf. Syst.*, 43
 546 (2):42:1–42:55, 2025.

547

548 Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open
 549 domain question answering. In *Proceedings of the 16th Conference of the European Chapter of
 550 the Association for Computational Linguistics: Main Volume*, pp. 874–880, 2021.

551

552 Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park. Adaptive-rag: Learning
 553 to adapt retrieval-augmented large language models through question complexity. In *Proceedings
 554 of the Conference of the North American Chapter of the Association for Computational Linguistics:
 555 Human Language Technologies (Volume 1: Long Papers)*, pp. 7036–7050, 2024.

556

557 Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *17th European
 558 Conference on Machine Learning*, volume 4212 of *Lecture Notes in Computer Science*, pp. 282–
 559 293, 2006.

560

561 Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal,
 562 Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe
 563 Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural
 564 Information Processing Systems 33: Annual Conference on Neural Information Processing Systems*,
 565 2020.

566

567 Yongjian Li, HaoCheng Chu, Yukun Yan, Zhenghao Liu, Shi Yu, Zheni Zeng, Ruobing Wang, Sen
 568 Song, Zhiyuan Liu, and Maosong Sun. Kare-rag: Knowledge-aware refinement and enhancement
 569 for rag. *arXiv preprint arXiv:2506.02503*, 2025.

570

571 Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. Query rewriting in retrieval-
 572 augmented large language models. In *Proceedings of the Conference on Empirical Methods in
 573 Natural Language Processing*, pp. 5303–5315, 2023.

574

575 Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon,
 576 Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder,
 577 Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative
 578 refinement with self-feedback. In *Advances in Neural Information Processing Systems 36: Annual
 579 Conference on Neural Information Processing Systems*, 2023.

580

581 Zachary McGraw. What triggered the google cloud outage
 582 in june 2025? [https://www.vcsolutions.com/blog/
 583 google-cloud-outage-causes-behind-the-june-2025-incident/](https://www.vcsolutions.com/blog/google-cloud-outage-causes-behind-the-june-2025-incident/), 2025.
 584 Last accessed: 2025-09-15.

585

586 S McKenzie, A Burgess, and C Mellis. Interns reflect: the effect of formative assessment with
 587 feedback during pre-internship. *Advances in Medical Education and Practice*, 8:51–56, 2017.

588

589 Kevin Meng, Arnab Sen Sharma, Alex J. Andonian, Yonatan Belinkov, and David Bau. Mass-editing
 590 memory in a transformer. In *The Eleventh International Conference on Learning Representations*,
 2023.

591

592 Neural Bridge AI. Retrieval-augmented generation (rag) dataset 1200. <https://huggingface.co/datasets/neural-bridge/rag-dataset-1200>, 2024a.

593

594 Neural Bridge AI. Retrieval-augmented generation (rag) dataset 12000. <https://huggingface.co/datasets/neural-bridge/rag-dataset-12000>, 2024b.

595

596 Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize
 597 Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. Chatdev:
 598 Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of
 599 the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15174–15186, 2024.

594 Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions
 595 for machine comprehension of text. In *Proceedings of the Conference on Empirical Methods in*
 596 *Natural Language Processing*, pp. 2383–2392, 2016.

597

598 Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion:
 599 language agents with verbal reinforcement learning. In *Advances in Neural Information Processing*
 600 *Systems 36: Annual Conference on Neural Information Processing Systems*, 2023.

601 Abraham Silberschatz, F. Korth Henry, and Shashank Sudarshan. *Database system concepts*. McGraw-
 602 Hill, New York, USA, 2002.

603

604 Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay
 605 Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cris-
 606 tian Canton Ferrer, Moya Chen, Guillem Cucurull, et al. Llama 2: Open foundation and fine-tuned
 607 chat models. *arXiv preprint arXiv:2307.09288*, 2023.

608 Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O’Sullivan, and
 609 Hoang D. Nguyen. Multi-agent collaboration mechanisms: A survey of llms. *arXiv preprint*
 610 *arXiv:2501.06322*, 2025.

611 Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman.
 612 GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th*
 613 *International Conference on Learning Representations*, 2019.

614

615 Fei Wang, Xingchen Wan, Ruoxi Sun, Jiefeng Chen, and Sercan Ö. Arik. Astute RAG: overcoming
 616 imperfect retrieval augmentation and knowledge conflicts for large language models. In *Proceed-
 617 ings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long
 618 Papers)*. Association for Computational Linguistics, 2025.

619 Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai
 620 Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. A survey on large
 621 language model based autonomous agents. *Frontiers Comput. Sci.*, 18(6):186345, 2024.

622

623 Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe
 624 Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou,
 625 Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxi-
 626 ang Weng, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huang, Qi Zhang, and Tao Gui.
 627 The rise and potential of large language model based agents: a survey. *Sci. China Inf. Sci.*, 68(2),
 628 2025.

629

630 Fangyuan Xu, Weijia Shi, and Eunsol Choi. RECOMP: improving retrieval-augmented lms with
 631 context compression and selective augmentation. In *The Twelfth International Conference on*
 632 *Learning Representations*, 2024.

633 LIN C. Y. Rouge : A package for automatic evaluation of summaries. *Proc. Workshop on Text
 634 Summariation Branches Out, Post-Conference Workshop of ACL 2004*, 2004.

635 Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. Corrective retrieval augmented generation.
 636 *arXiv preprint arXiv:2401.15884*, 2024.

637

638 Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan.
 639 Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural*
 640 *Information Processing Systems 36: Annual Conference on Neural Information Processing Systems*,
 641 2023a.

642

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

646

647 Wenhao Yu, Hongming Zhang, Xiaoman Pan, Peixin Cao, Kaixin Ma, Jian Li, Hongwei Wang,
 648 and Dong Yu. Chain-of-note: Enhancing robustness in retrieval-augmented language models. In
 649 *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp.
 650 14672–14685, 2024.

648 Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi,
649 Shengyu Mao, Jintian Zhang, Yuansheng Ni, Siyuan Cheng, Ziwen Xu, Xin Xu, Jia-Chen Gu,
650 Yong Jiang, Pengjun Xie, Fei Huang, Lei Liang, Zhiqiang Zhang, Xiaowei Zhu, Jun Zhou, and
651 Huajun Chen. A comprehensive study of knowledge editing for large language models. *arXiv*
652 preprint *arXiv:2401.01286*, 2024.

653 Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao,
654 Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi.
655 Siren’s song in the AI ocean: A survey on hallucination in large language models. *arXiv preprint*
656 *arXiv:2309.01219*, 2023.

657 Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H. Chi, Quoc V. Le,
658 and Denny Zhou. Take a step back: Evoking reasoning via abstraction in large language models.
659 In *The Twelfth International Conference on Learning Representations*, 2024.

660 Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang,
661 Zi Lin, Zhuchan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica.
662 Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information
663 Processing Systems 36: Annual Conference on Neural Information Processing Systems*, 2023.

664

665

666

667

668

669

670

671

672

673

674

675

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

702 APPENDIX
703704 A SAMPLE WITH EACH ERROR TYPE
705706 Shared Context for All Examples
707

708 • **User Query (Q):** *Please describe the key identifying features of the Death Cap mushroom.*
709

710 • **Oracle Knowledge Chunk (K_{oracle}):** ...*The Death Cap (*Amanita phalloides*) is one of the world's*
711 *most poisonous basidiomycetes. It contains α -amanitin, which causes fatal liver and kidney*
712 *damage. The species typically grows in summer and autumn in deciduous and mixed forests,*
713 *forming mycorrhizal relationships with broadleaf trees like oaks. Its key identifying features*
714 *include: a cap that is typically pale grey, yellowish-green, or olive-green, and smooth. The gills are*
715 *white and free (i.e., not attached directly to the stipe). The most important feature is that the base*
716 *of its stipe has a distinct, cup-like volva, and the upper part of the stipe usually has an annulus*
717 *(ring)...*

718 • **Correct Answer (A_{oracle}):** *The key features of the Death Cap are: 1) A cap color of pale grey,*
719 *yellowish-green, or olive-green; 2) Gills that are white and free; and 3) A cup-like volva structure*
720 *at the base of the stipe.*

721 FULLY INCORRECT
722

723 • **Erroneous Knowledge Chunk (K):** ...*The Death Cap (*Amanita phalloides*) is one of the world's*
724 *most poisonous basidiomycetes. It contains α -amanitin, which causes fatal liver and kidney*
725 *damage. The species typically grows in summer and autumn in deciduous and mixed forests. Its*
726 *key identifying features include: a cap that is typically bright red with white warts. The gills are*
727 *yellow and decurrent (i.e., running down the stipe). The base of its stipe lacks any special cup-like*
728 *structure, simply tapering at the base, and the upper part of the stipe usually has an annulus*
729 *(ring)...*

730 • **Generated Answer from K (A):** *The key features of the Death Cap include: 1) A cap that is bright*
731 *red with white warts; 2) Gills that are yellow and decurrent; and 3) No cup-like structure at the*
732 *base of its stipe.*

733 PARTIALLY INCORRECT
734

735 • **Erroneous Knowledge Chunk (K):** ...*The Death Cap (*Amanita phalloides*) is one of the world's*
736 *most poisonous basidiomycetes. It contains α -amanitin, which causes fatal liver and kidney*
737 *damage. The species typically grows in summer and autumn in deciduous and mixed forests. Its*
738 *key identifying features include: a cap that is typically pure white or light brown, and smooth. The*
739 *gills are white and free (i.e., not attached directly to the stipe). The most important feature is that*
740 *the base of its stipe has a distinct, cup-like volva, and the upper part of the stipe usually has an*
741 *annulus (ring)...*

742 • **Generated Answer from K (A):** *The key features of the Death Cap include: 1) A cap color of*
743 *pure white or light brown; 2) Gills that are white and free; and 3) A cup-like volva structure at the*
744 *base of the stipe.*

745 FULLY MISSING
746

747 • **Erroneous Knowledge Chunk (K):** ...*The Death Cap (*Amanita phalloides*) is one of the world's*
748 *most poisonous basidiomycetes and is responsible for the vast majority of fatal mushroom poi-*
749 *sonings worldwide. Its primary toxin is α -amanitin, a heat-stable cyclopeptide that cannot be*
750 *destroyed by cooking. Symptoms of poisoning typically do not appear until 6 to 12 hours after*
751 *consumption, starting with severe abdominal pain and vomiting. This is followed by a brief period*
752 *of false recovery, which ultimately leads to catastrophic liver and kidney failure...*

753 • **Generated Answer from K (A):** *Regarding the Death Cap mushroom, the provided information*
754 *indicates that it is extremely toxic and causes the majority of fatal mushroom poisonings. Its toxin,*
755 *α -amanitin, is heat-stable, and symptoms of poisoning are delayed, leading to severe liver and*
756 *kidney failure.*

756

PARTIALLY MISSING

757

- **Erroneous Knowledge Chunk (K):** ...*The Death Cap (Amanita phalloides) is one of the world's most poisonous basidiomycetes. It contains α -amanitin, which causes fatal liver and kidney damage. The species typically grows in summer and autumn in deciduous and mixed forests. Its key identifying features include: a cap that is typically pale grey, yellowish-green, or olive-green, and smooth. The gills are light brown and decurrent (i.e., running down the stipe). The upper part of the stipe usually has an annulus (ring)...*
- **Generated Answer from K (A):** *The key features of the Death Cap include: 1) A cap color of pale grey, yellowish-green, or olive-green; and 2) Gills that are white and free.*

766

MIXED

767

- **Erroneous Knowledge Chunk (K):** ...*The Death Cap (Amanita phalloides) is one of the world's most poisonous basidiomycetes. It contains α -amanitin, which causes fatal liver and kidney damage. The species typically grows in summer and autumn in deciduous and mixed forests. Its key identifying features include: a cap that is typically pale grey, yellowish-green, or olive-green, and smooth. The gills are light brown and decurrent (i.e., running down the stipe). The upper part of the stipe usually has an annulus (ring)...*
- **Generated Answer from K (A):** *The key features of the Death Cap include: 1) A cap color of pale grey, yellowish-green, or olive-green; and 2) Gills that are light brown and decurrent.*

776

777

ConvCorrect-SQ		ConvCorrect-NB	
Statistic	Value	Statistic	Value
Total Samples		Total Samples	
124,531		80,772	
Perturbation Type Distribution		Perturbation Type Distribution	
Replace	32.8%	Replace	31.8%
Add	33.5%	Add	34.8%
Delete	33.7%	Delete	33.4%
Error Type Distribution		Error Type Distribution	
Fully incorrect	77.4%	Fully incorrect	55.5%
Partially incorrect	11.9%	Partially incorrect	29.7%
Fully missing	0.2%	Fully missing	0.5%
Partially missing	10.1%	Partially missing	11.4%
Mixed	0.4%	Mixed	2.9%

793

Table 5: Key statistics of the ConvCorrect Benchmark, detailing the scale and distribution of types across both splits.

794

795

796

797

798

799

800

801

802

803

804

805

806

807

808

809

810 B PROMPTS FOR MULTI-AGENT SYSTEM
811812 This section provides the detailed prompts used for each agent in the MT-KOC framework. These
813 prompts are presented with maximum fidelity to the implementation code to ensure reproducibility.
814 Each major prompt is presented on a separate page for clarity. Placeholders are denoted with angle
815 brackets (e.g., <query>).
816817 B.1 PROMPT FOR KNOWLEDGE DISTILLATION AGENT
818819 The agent is prompted to act as a "Factual Extractor." It uses the user's query and the erroneous
820 Knowledge Chunk as a hint to find a minimal, direct, and factual piece of reference information.
821822 You are a hyper-focused Factual Extractor. You will be
823 given a [Query] and an [Erroneous Knowledge Chunk]. The
824 [Erroneous Knowledge Chunk] is a previous, flawed attempt to
825 answer the query. Your task is to use this erroneous chunk
826 as a crucial hint to disambiguate the user's true intent.
827 You must adhere to the following strict rules:
828829 1. **Use Knowledge Chunk for Disambiguation:** Analyze the
830 [Erroneous Knowledge Chunk] to understand the specific
831 angle or meaning the user is interested in, especially
832 for ambiguous queries.
833 2. **Direct Answer First:** Primary goal is to find a
834 direct, minimal, and factual answer to the [Query],
835 informed by the hint from the [Erroneous Knowledge
836 Chunk]. Extract only the core piece of information
837 needed. For example, if asked for a capital city,
838 provide only the city's name. If asked for a date,
839 provide only the date.
840 3. **Strict Relevance Filter:** Aggressively filter out
841 any information that is not essential to answering
842 the query. Ignore historical context, biographical
843 details, related trivia, or explanations unless the
844 query explicitly asks for them.
845 4. **Mandatory Generative Fallback:** If, and only if, no
846 verifiable information is found, you **must** generate a
847 plausible and concise answer, prefaced with: "The
848 following information is generated for reference only
849 and may not be accurate, as no verified data was found
850 in the internal knowledge base."
851 5. **No Empty Responses:** You must always provide a
852 response.
853 6. **Format:** Return the information as plain text.854 **Examples:**
855 [Query]: Who was the first person to walk on the moon?
856 [Erroneous Knowledge Chunk]: The first person to walk on
857 the moon was Yuri Gagarin.
858 [Output]: Neil Armstrong.
859 [Query]: What is Java?
860 [Erroneous Knowledge Chunk]: Java is a popular caffeinated
861 beverage made from roasted coffee beans.
862 [Output]: Java is a large island in Indonesia.
863

864 B.2 PROMPT FOR ACTION RECOMMENDATION AGENT
865866 The agent is prompted to analyze the user feedback, the potentially erroneous Knowledge Chunk,
867 and the reference information to generate a set of structured correction hypotheses (actions).
868869 You are a brilliant AI Recommendation Generator. Your
870 task is to analyze user feedback and a potentially flawed
871 [Knowledge Chunk] to generate plausible correction
872 hypotheses. You will be given [Reference Information] to
873 assist you, but you must use it critically.
874**Analysis Protocol:**

1. **Prioritize User Feedback:** Your primary guide is always the user's [Feedback].
 - **Specific Feedback:** If the [Feedback] is direct and clear (e.g., "The capital is Paris, not Berlin"), your main goal is to create a recommendation that directly implements this feedback. The [Reference Information] should be considered secondary, used only to verify or add minor, consistent details. The user's explicit correction takes precedence.
 - **Vague Feedback:** If the [Feedback] is vague (e.g., "That's wrong," "It's incomplete"), you must critically use the [Reference Information] as your main tool to deduce the user's intent. Compare the [Knowledge Chunk] with the [Reference Information] to find likely errors or omissions. Generate 2-4 distinct hypotheses based on these potential discrepancies. Treat the reference as a strong hint, not an absolute truth.
2. **Recommendation Structure:** Each recommendation must be a self-contained object with two keys: '"description"' and '"action"'.
3. **Action Mapping:** The "action" object must contain: '"action_type"' (one of ["REVISE", "ADD", "DELETE"]) and '"recommendation"'.
4. **Output Format:** Return ONLY a valid JSON array of recommendation objects. If no corrections seem necessary, return an empty JSON array '[]'.

Input Format:

[Query]: The user's original question.
 [Previous response]: The model's previous answer.
 [Knowledge Chunk]: The original Knowledge Chunk that needs correction.
 [Feedback]: The user's feedback.
 [Reference Information]: Factual information provided for inspiration and guidance. **Note: This information is for reference only and is not guaranteed to be the absolute truth.**

918 B.3 PROMPTS FOR CORRECTION AGENT
919920 The agent acts as a precise text editor, executing a single action provided by the Action Recommen-
921 dation Agent. The following pages detail the prompts for each of its modules.
922923 ADD MODULE
924925 You are a precise AI text editor. Your sole task is to
926 add information to the [Knowledge Chunk] based on a single,
927 clear [ADD Instruction].
928**Instructions:**

1. **Identify Information:** Read the [ADD Instruction] to understand what new information needs to be added.
2. **Execute Insertion:** Insert the new information into the most logical and natural position within the [Knowledge Chunk].
3. **Strict Adherence:** You **MUST** add the information exactly as provided in the instruction. Do not alter it.
4. **Preserve Everything Else:** All other parts of the original Knowledge Chunk must be retained exactly.
5. **Output Plain Text:** Output only the complete, modified Knowledge Chunk as plain text.

Example:

[Knowledge Chunk]: Japan is an island nation in East Asia.
 [ADD Instruction]: Add 'with a population of approximately 125 million' to the description of Japan.
 [Output]: Japan is an island nation in East Asia with a population of approximately 125 million.

948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971

972 REVISE MODULE

973

974

975 You are a precise AI text editor. Your sole task is to
976 modify the [Knowledge Chunk] based on a single, clear
977 [REVISE Instruction].

978

Instructions:

979

1. **Identify Target:** Read the [REVISE Instruction] to understand which part of the [Knowledge Chunk] is incorrect.
2. **Execute Correction:** Revise the incorrect part with the new information provided in the [REVISE Instruction].
3. **Strict Adherence:** You **MUST** use the information exactly as given in the instruction. Do not add, infer, or hallucinate any information not present in the instruction.
4. **Preserve Everything Else:** All other parts of the Knowledge Chunk must be retained exactly.
5. **Output Plain Text:** Output only the complete, modified Knowledge Chunk as plain text.

980

981

982

983

984

985

986

987

988

989

990

991

992

993

Example:

994

995 [Knowledge Chunk]: The capital of France is Florida, a
996 vibrant city known for its art museums. France is in
997 Europe.

998

999 [REVISE Instruction]: In the Knowledge Chunk, change the
1000 capital of France from 'Florida' to 'Paris'.

1001

1002 [Output]: The capital of France is Paris, a vibrant city
1003 known for its art museums. France is in Europe.

1004

1005

1006

1007

1008

1009

1010

1011

1012

1013

1014

1015

1016

1017

1018

1019

1020

1021

1022

1023

1024

1025

1026 DELETE MODULE

1027

1028

1029 You are a precise AI text editor. Your sole task is to
 1030 remove a misleading segment from the [Knowledge Chunk] based
 1031 on a single, clear [DELETE Instruction].

1032 **Instructions:**

- 1033 1. **Identify Target:** Read the [DELETE Instruction] to
 1034 understand which specific part of the Knowledge Chunk
 1035 is misleading and should be removed.
- 1036 2. **Execute Deletion:** Remove only the identified
 1037 misleading segment from the [Knowledge Chunk].
- 1038 3. **Preserve Everything Else:** All other parts of the
 1039 Knowledge Chunk must be retained exactly.
- 1040 4. **Ensure Coherence:** The remaining text must be
 1041 grammatically correct and coherent.
- 1042 5. **Output Plain Text:** Output only the complete, modified
 1043 Knowledge Chunk as plain text.

1044 **Example:**

1045 [Knowledge Chunk]: The largest planet is Jupiter, which is
 1046 a star and has many moons. It orbits the Sun.

1047 [DELETE Instruction]: Remove the phrase ', which is a
 1048 star,' as it incorrectly describes Jupiter.

1049 [Output]: The largest planet is Jupiter, which has many
 1050 moons. It orbits the Sun.

1051

1052

1053

1054

1055

1056

1057

1058

1059

1060

1061

1062

1063

1064

1065

1066

1067

1068

1069

1070

1071

1072

1073

1074

1075

1076

1077

1078

1079

1080 B.4 PROMPTS FOR REWARD EVALUATION AGENT
10811082 The agent is composed of two modules: one for generating an answer and one for scoring it. The
1083 following pages detail the prompts for each module.
10841085 ANSWER GENERATION MODULE
10861087 You are a knowledgeable AI assistant that interacts
1088 with users through natural conversation. Follow these
1089 guidelines:
1090

1. Treat the provided Knowledge Chunk as normal and complete, and base your responses strictly on it. If the Knowledge Chunk lacks relevant details or is contradictory, generate a natural and concise answer using available information or reasonable inference.
2. Use accurate terminology and proper names from the Knowledge Chunk, paraphrasing naturally for clear and conversational responses.
3. Keep responses concise, addressing the core of the question in 1-2 sentences, focusing only on the most relevant information.
4. Present information in a fluent, natural conversation style, avoiding phrases like 'the information provided' or references to external sources. For missing information, provide a plausible answer based on the Knowledge Chunk or inference.
5. If the Knowledge Chunk contains contradictory information, include all relevant details in the answer while maintaining natural expression.
6. Always attempt to answer the question, even if the Knowledge Chunk lacks direct information, by leveraging related content or reasonable assumptions.

1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133

1134
1135

SCORING MODULE

1136

1137 You are a strict and objective Answer Evaluator. Your
 1138 sole purpose is to score a new answer based on its precise
 1139 alignment with a ground truth. Your evaluation must be
 1140 rigorous and prioritize factual accuracy over stylistic
 1141 qualities.

1142

Evaluation Protocol (Strict Priority Order):

1143

1. **Determine the Ground Truth:** You must determine the
 1144 ground truth by following this hierarchy:

1145

- **Tier 1: Clear Feedback:** If the [User Feedback] is specific and provides a clear correction (e.g., "it should be Paris"), that feedback **is the absolute and sole ground truth**. You MUST ignore the [Reference Information], even if it contains additional or conflicting details.

1146

- **Tier 2: Vague Feedback with Reference:** If the [User Feedback] is vague (e.g., "it's wrong") **AND** the [Reference Information] is available, then the [Reference Information] **becomes the ground truth**.

1147

- **Tier 3: Vague Feedback, No Reference (Inference):** If the [User Feedback] is vague **AND** the [Reference Information] is empty or unavailable, you must first use your own general knowledge to determine what a perfect answer to the [Query] would be. This **inferred ideal answer** then becomes your ground truth for the evaluation.

1148

2. **Score based on Ground Truth Alignment (0-10):**

1149

- **10 (Perfect Match):** The new answer perfectly incorporates the ground truth. All key entities, facts, and numbers from the ground truth are present and correct.

1150

- **7-9 (High Alignment):** The new answer correctly incorporates the main point of the ground truth but may miss a minor detail.

1151

- **4-6 (Partial Alignment):** The new answer addresses the ground truth partially (e.g., corrects one error but misses another).

1152

- **0-3 (Low Alignment):** The new answer attempts to address the ground truth but fails significantly, containing major inaccuracies.

1153

Scoring Rules:

1154

- **Precision is Key:** Focus entirely on the presence and correctness of key information from the ground truth you determined in Step 1.

1155

- **Output ONLY the numerical score (0-10).**

1156

1157

1158

1159

1160

1161

1162

1163

1164

1165

1166

1167

1168

1169

1170

1171

1172

1173

1174

1175

1188 C IMPLEMENTATION DETAILS
11891190 To ensure a fair and controlled comparison, all methods evaluated in our experiments, including all
1191 baselines and every agent within the MT-KOC framework, are powered by the same underlying large
1192 language model.
11931194 **Base Model and Environment.** We use **DeepSeek-R1-Distill-Qwen-32B** as the base model for
1195 all experiments. The model is deployed on 8 H100 GPUs using the VLLM framework for efficient
1196 inference.
11971198 **MT-KOC Hyperparameters.** For our proposed MT-KOC framework, the number of search epochs
1199 is set to 8. The exploration coefficient E in the UCT-based selection metric (Eq. 6) is set to 1.3. The
1200 Shortcut mechanism is triggered when an evaluation score S_i from the Reward Evaluation Agent
1201 reaches the maximum value of 10.0.
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241