RAG Knowledge Online Correction with Conversation-Based User Feedback

Anonymous ACL submission

Abstract

Retrieval-Augmented Generation (RAG) is a promising paradigm to enhance LLMs by integrating external knowledge. However, its performance degrades when the knowledge contains errors. When users encounter such errors, the typical correction process involves users reporting the errors, after which service providers investigate the knowledge base to identify and fix the issues. This process is often time-consuming, and in the meantime, other users continue to encounter the same errors, leading to a poor user experience. To address this challenge, we propose a new task, **Knowledge Online Correction**, which focuses on correcting errors immediately after they are pointed out by users through conversation-based feedback. To evaluate this task, we conducted a preliminary user study and, based on the findings, developed a new benchmark, **ConvCorrect**. To address this task, we propose **MT-KOC** (Multi-step Knowledge Online Correction), a multi-agent framework that utilizes a dynamic action search algorithm to identify the optimal correction sequence for progressive correction. Empirical results on ConvCorrect show that our method significantly outperforms existing baselines. Our code is available at https://anonymous.4open.science/r/MT-KOC.

1 Introduction

Large Language Models (LLMs) have achieved remarkable success in real-world applications, but still face challenges such as hallucination and outdated knowledge (Huang et al., 2025; Zhang et al., 2023a). Retrieval-Augmented Generation (RAG) has emerged as a promising solution by integrating external knowledge to mitigate these issues (Lewis et al., 2020; Gao et al., 2023b). However, its performance degrades when the knowledge base contains errors (e.g., Fig. 1A). When users encounter such errors, the typical correction process involves tedious manual intervention: users report the issue, and service providers investigate the knowledge base to identify and correct the errors (Fig. 1B). This process is often time-consuming, taking several hours to multiple working days (McGraw, 2025). During this period, the knowledge errors persist and continue to mislead users, resulting in a poor overall experience.

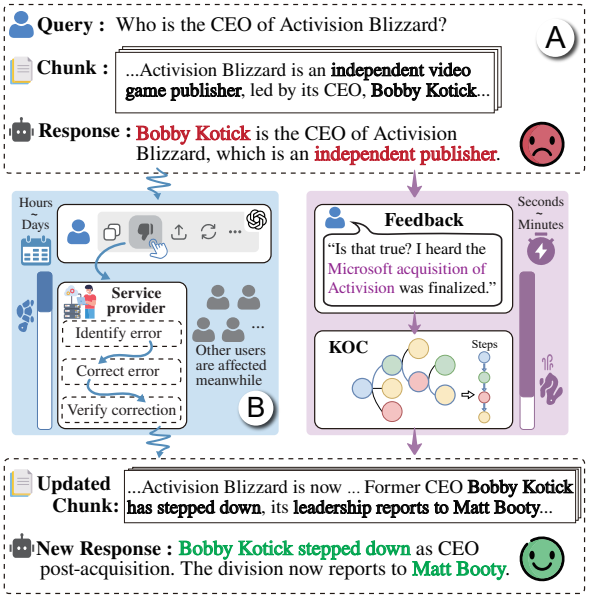


Figure 1: Comparison between slow manual correction (left) and rapid knowledge online correction (right).

To address the issue of inefficiency, several methods have been proposed (Yan et al., 2024; Asai et al., 2024). These methods automatically correct the errors online through web retrieval (e.g., Wikipedia). However, the effectiveness of these methods depends on the accuracy of the web retrieval results. If the web retrieval results are incorrect, these methods have no alternative mechanism to resolve the errors. Most recently, STACKFEED (Gupta et al., 2024) leverages external feedback (e.g., compiler errors) to further enhance the accuracy of error corrections. However, STACKFEED works only when sufficient feedback is collected, which fails to meet the requirements of online correction.

To tackle these issues, we drew inspiration from the **Formative Assessment with Feedback** model in education (McKenzie et al., 2017). In this model, teachers provide 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 cor-

rected. 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 feedback to correct knowledge errors within the same conversational session.

While this new strategy has the potential to help correct knowledge errors, 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. **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 develop 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 and how users provide feedback for these errors. Based on this summarization, we synthesize errors in the knowledge and the conversation-based feedback, which were 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 MT-KOC outperforms prior SOTA by an average of 5.6 points in 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, an online correction method that validates user feedback and uses a dynamic action search algorithm to correct the knowledge base.
3. We empirically validate MT-KOC, demonstrating its effectiveness through extensive experiments, which show an improvement over existing SOTA methods.

2 Related Work

Retrieval-Augmented Generation (RAG) mitigates LLM hallucinations by grounding generation in external knowledge (Brown et al., 2020; Chowdhery et al., 2022; Touvron et al., 2023; Lewis et al., 2020). Typical research along this line includes query formulation (Zheng et al., 2024; Ma et al., 2023; He et al., 2023), adaptive retrieval (Asai et al., 2024; Jeong et al., 2024), context refinement (Yu et al., 2024; Li et al., 2025; Xu et al., 2024; Zhang et al., 2023b) and post-generation verification (Li and Flanigan, 2025). However, these methods typically assume the knowledge contains no errors, which is not always true in real-world applications. Therefore, many methods have been proposed to address knowledge errors, which generally fall into two categories: answer and knowledge correction.

Answer correction methods directly revise the answers of LLMs when encountering knowledge errors without modifying the knowledge. For example, ReEx (Kim et al., 2024) retrieves external evidence to detect, explain, and revise factual errors in answers. Purr (Chen et al., 2023) denoises hallucinations by detecting text corruptions in answers. Syncheck (Wu et al., 2024) monitors faithfulness during answer generation and revises them when necessary. Although effective, these methods cannot revise the errors in the knowledge, which will mislead other users.

Knowledge correction methods directly revise the errors in the knowledge base. The majority of these methods correct the knowledge base without leveraging any external feedback. Typical strategies include rewriting retrieved contexts (Qiao et al., 2025; Dong et al., 2025), retrieving supplementary information (Yan et al., 2024), resolving conflicts between internal and external knowledge (Wang et al., 2025; Gao et al., 2023a), merging generated and retrieved passages via compatibility scoring (Zhang et al., 2023b), or pruning redundant facts using entropy for multi-hop QA (Shi et al., 2024). Recently, STACKFEED (Gupta et al., 2024) is developed to leverage external feedback for better knowledge correction. It utilizes a reinforcement learning strategy to revise the knowledge base from a batch of external feedback (e.g., compiler errors). However, STACKFEED requires a substantial amount of feedback to be collected, failing to meet the online requirements of KOC, which demands error correction within seconds or minutes. In contrast, our method can correct the knowledge base after each single user feedback is received.

3 Benchmark

3.1 Problem Formulation

Given a user query Q, a knowledge chunk K is retrieved, which includes errors. Based on the chunk K, an answer A is generated by an LLM: $A = \text{LLM}(Q, K)$. After receiving the answer, the user responds with conversation-based feedback F. Then, the goal of the KOC task is to find a transformation function T such that the answer generated based on the chunk K^* transformed by T is similar to the answer generated based on the correct chunk K_{oracle}:

$$T^* = \underset{T}{\text{argmin}} \quad \text{Diff}\{\text{LLM}(Q, K^*), \text{LLM}(Q, K_{\text{oracle}})\}$$
$$\text{s.t.} \quad K^* = T(K, F) \tag{1}$$

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

3.2 Preliminary User Study

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

Study setup. To ensure the representativeness of the user study participants, we employed a hybrid participant pool consisting of 12 graduate students and 20 novices. The 12 graduate students majoring in Computer Science, aged from 21 to 29 years (mean = 24.25, SD = 2.60). All of them have more than one year of experience in developing or using RAG systems. Upon completion, each participant received a $10 compensation, independent of their performance. The 20 novices were recruited from Amazon Mechanical Turk. We did not restrict the background of the recruited workers because we wanted select representative ones from the potential audiences. However, to ensure the workers took these tasks seriously, the selected workers should have completed at least 1,000 HITs with approval rate greater than 98%. Each worker received a $6 compensation for their participation.

Datasets. The study was constructed on 2,400 samples randomly selected from the Neural-Bridge Dataset (Neural Bridge AI, 2024b,a). Each sample contains a query Q and a correct knowledge chunk K_{oracle}. Then, the chunk of each sample was perturbed to contain errors. To ensure the errors encompass most real-world scenarios, the perturbations are capable of generating any type of error. In the field of databases, updating, adding, and removing are used to cover all types of modifications. Motivated by this, we also include three types of perturbations (Silberschatz et al., 2002). (1) *Update:* A key entity (e.g., date, name, location) in the relevant section is updated to an incorrect but plausible entity of the same type. (2) *Add:* A contextually coherent but factually incorrect or misleading sentence is added after the sentence containing the relevant information. (3) *Delete:* A critical phrase or sentence required to answer the query is removed from the relevant section. The chunk of each sample was randomly applied one of the three types of perturbations, resulting in an erroneous knowledge chunk (K). Based on K, an answer (A) was generated by an LLM.

Procedure. The 2,400 samples were randomly distributed to the 12 graduate students and 20 novices. Each graduate student received 100 samples, while each novice was assigned 60 samples. For each sample, the participants were asked to assume the role of a user who had received an erroneous answer (A) from a conversational AI system after posing a query. Their tasks were to write a conversation-based feedback (F) in response to the erroneous answer, reflecting what they would realistically provide in such a scenario. The full instructions provided to the participants are detailed in Appendix B.

Error type. After the user study, two researchers from our team were tasked with classifying the error type of the answer (A) in each of the 2,400 samples. Each researcher was responsible for half of the data, and a cross-review was conducted upon completion to ensure consistency. This process resulted in the classification of all samples into five primary error types: *Fully Incorrect*, *Partially Incorrect*, *Fully Missing*, *Partially Missing*, and *Mixed*. For illustrative examples of each error type, see Appendix A.

Error Type	Associated Feedback Styles
Fully incorrect	Direct correction, Error indication
Partially incorrect	Direct correction, Error indication, Error localization
Fully missing	Direct completion, Missing indication
Partially missing	Direct completion, Missing indication
Mixed	Direct correction and completion, Error and missing indication
All Types	Abnormal feedback

Table 1: Mapping from Error Types to typical, **normal** Feedback Styles. The final row indicates a small subset of non-cooperative, **abnormal** feedback observed across all error types.

Typical conversation-based feedback styles. After the user study was completed, the same two researchers analyzed the feedback (F) provided by the participants. Following a similar process of independent annotation and cross-review, they categorized the majority of the collected feedback into seven types of *normal feedback* and one type of *abnormal feedback*. These types of feedback are summarized in Table 1.

Normal feedback. These are the types of feedback users are expected to provide, offering additional information to help correct knowledge errors. For example, some participants offered a *direct correction* (e.g., "No, the director was James Cameron"), while others only gave an *error indication* without correction (e.g., "That's not the right director"), or pointed to missing information with a *missing indication* (e.g., "You forgot to mention the main actor").

Abnormal feedback. This type of feedback includes the ones clearly malicious (e.g., insults), intentionally misleading (providing new incorrect information), or entirely irrelevant to the query. Our statistical analysis, **detailed in Appendix Table 7**, indicated that such feedback accounted for 10.58% (254 of 2,400) of the total responses. As this type of feedback was not correlated with any specific error type, it is classified as a univer-

sal style applicable to all error types in our taxonomy (Table 1, final row).

From Table 1, a key finding is that the relationship between error types and feedback styles is a complex one-to-many mapping. This discovery highlights a fundamental challenge: a simple, low-level textual perturbation can manifest as a wide spectrum of high-level, user-perceived semantic errors. This non-trivial mapping proves that a simplistic approach(e.g., mechanically generating feedback based on the perturbation type) would be unrealistic and fail to capture the task's true complexity. Therefore, this empirically derived taxonomy not only reveals the difficulty of the task, but also serves as the principal blueprint for our benchmark construction.

3.3 Benchmark Construction

Based on the error types and typical conversation-based feedback styles identified in the preliminary user study, we developed the first benchmark for the KOC task, ConvCorrect, with the help of LLMs. Throughout the process, we used GPT-5.1 as the base LLM due to its strong performance.

Data Source and Pre-processing. We construct ConvCorrect upon three diverse RAG benchmarks: SQuAD (Rajpurkar et al., 2016), Neural-Bridge (Neural Bridge AI, 2024b,a), and HotpotQA (Yang et al., 2018), resulting in three splits: ConvCorrect-SQ, ConvCorrect-NB, and ConvCorrect-HP. We first pre-processed these source datasets to create a clean base for perturbation. For SQuAD, which features multiple questions per context, we retained only one unique sample per context, merging the training and test sets to yield 20,958 initial samples. For Neural-Bridge, we combined its two versions (RAG-Dataset-12000 & RAG-Dataset-1200) and removed duplicates to obtain 12,893 unique samples. For HotpotQA, we utilized its validation set, which focuses on multi-hop question answering and consists of samples all designated as hard difficulty. After pre-processing, all datasets underwent sensitive content filtering. This resulted in a final set of 19,903 samples from SQuAD, 12,000 from Neural-Bridge, and 7,400 from HotpotQA, which formed the basis for our benchmark construction.

Construction Pipeline. Based on the pre-processed datasets, we constructed ConvCorrect in two main steps: **error generation** and **feedback generation**. The prompts used in this pipeline are provided in Appendix C.

- *Error generation.* Since the knowledge chunks in the source datasets are error-free, we first perturbed the chunks to contain errors. Similar to the preliminary user study, we consider three types of perturbations here. For each chunk, all three types of perturbations were applied using LLMs (Ding et al., 2024), resulting in three distinct perturbed versions. We then employed an LLM-based evaluator to filter out invalid perturbations where the correct answer remained derivable or unrelated context was altered. These valid perturbed chunks were then classified into the five error types identified in the preliminary user study using an LLM, and the accuracy of these classifications was subsequently verified by two researchers on a randomly sampled 10% subset, achieving 98.3% accuracy.

- *Feedback generation.* To ensure the diversity and richness of the feedback while maintaining the efficiency of the generation process, we employed a semi-automated method to create the feedback. We first invited ten experts to write the feedback for a 5% subset, which covers all the error types and feedback styles. Then, for the remaining samples, we provided the LLM with a randomly selected human-written example with the same error type and feedback style as in-context demonstrations. The LLM, acting as a **User**, was then prompted to generate new feedback messages (F) in the corresponding style. Since our user study found that around 10% of the feedback was abnormal, we applied an additional generation step for a random 10% of the samples to have abnormal feedback.

Dataset	Method	HD-D	SD	Validity (%)
Human (Reference)		0.7325	0.9176	100.0
ConvCorrect-SQ	Fully-automated	**0.7012**	**0.9027**	75.3
	Semi-automated	0.6619	0.8664	**96.5**
ConvCorrect-NB	Fully-automated	**0.6988**	**0.9005**	72.8
	Semi-automated	0.6890	0.8721	**95.2**
ConvCorrect-HP	Fully-automated	**0.7105**	**0.8837**	78.2
	Semi-automated	0.6964	0.8585	**97.1**

Table 2: Comparison of HD-D, SD and validity.

Quality Validation. To validate the effectiveness of the semi-automated construction process, we compared it with a fully-automated method and a human-written method. The fully-automated method is the same as our semi-automated method except that it does not introduce the human-written examples. Given the substantial time and cost involved, the human-written method only invite human to write feedback for a 5% subset. We utilized three metrics: Hypergeometric Distribution-based Divergence (HD-D) (Masry et al., 2025) to measure vocabulary richness, the Semantic Diversity (SD) score (Wang et al., 2023) to assess semantic variance, and generation validity. A feedback is considered invalid if it fails to provide useful information (i.e., misleading). The validity is evaluated by human on a 5% subset.

As shown in Table 2, the validity of our semi-automated method is higher than the fully-automated method and close to the human-written method. Moreover, our method demonstrates a comparable level of vocabulary and semantic diversity compared to the other two methods.

Statistics. The final dataset consists of **130,444** samples for ConvCorrect-SQ, **93,707** for ConvCorrect-NB,

and **49,130** for ConvCorrect-HP. The detailed statistics of the Benchmark's scale and type distribution are presented in Appendix Table 8.

4 Method

4.1 Framework Overview

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

4.2 Multi-Agent System

The multi-agent system consists of four collaborative agents: a **Knowledge Extraction Agent**, an **Action Recommendation Agent**, a **Correction Agent**, and a **Reward Evaluation Agent**. Implementation prompts for all agents are provided in Appendix D.

Knowledge Extraction Agent. Effective correction requires both grounding in reliable facts and validating user feedback. Accordingly, the agent operates in two sequential steps. The first performs a plausibility check on the user's feedback (F), assigning a 0–10 score based on intrinsic plausibility rather than parametric agreement, so plausibly novel feedback is not penalized. Implausible or malicious feedback falls below a predefined threshold.

The subsequent knowledge extraction step adapts based on this validation result. If the feedback passes validation, the agent utilizes both the feedback (F) and the original query (Q) to synthesize the reference information (I), capturing the user's corrective intent ($I = f_{\text{extract}}(Q, F)$). If the feedback fails validation, the agent excludes the feedback and synthesizes reference information based solely on the query (Q) and its internal parametric knowledge ($I = f_{\text{extract}}(Q)$). This mechanism prevents the incorporation of erroneous user inputs while maintaining the system's ability to provide correct information.

Action Recommendation Agent. As pointed out in a recent survey (Tran et al., 2025), the performance of LLMs would be improved if a complex action is decomposed into several sub-actions. Therefore, we also decompose the correction action into three commonly used sub-actions: `ADD`, `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 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 $\{C_1, ..., C_k\}$. Therefore, action recommendation agents recommends suitable actions given the user query (Q), current knowledge chunk (K), user feedback (F), and reference information (I).

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}:

$$K_{i+1} = f_{\text{correct}}(K_i, C_i) \tag{2}$$

Where f_{correct} denotes the execution logic.

Reward Evaluation Agent. This agent evaluates 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}:

$$S_{i+1} = f_{\text{eval}}(K_{i+1}, Q, F, I) \tag{3}$$

The evaluation function f_{eval} is implemented by the agent, which is prompted to act as an impartial judge. To validate this metric, we compared the agent's scores with human ratings on a random subset. The strong correlation confirms the reliability of our approach (details in Appendix E).

4.3 Dynamic Action Search

Based on the multi-agent system, it is required to identify the optimal sequence of 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:

$$\mathbf{C}^* = \arg\max_{\mathbf{C}} \quad \mathbb{E}\left[f_{\text{eval}}\left(\text{Apply}(K_0, \mathbf{C}), Q, F, I\right)\right] \tag{4}$$

1. Initialization. The process begins with the initial erroneous Knowledge Chunk, K_0, as the root node. The Knowledge Extraction Agent is first invoked to generate the global Reference Information (I). This step applies the validation logic: if the user feedback (F) passes

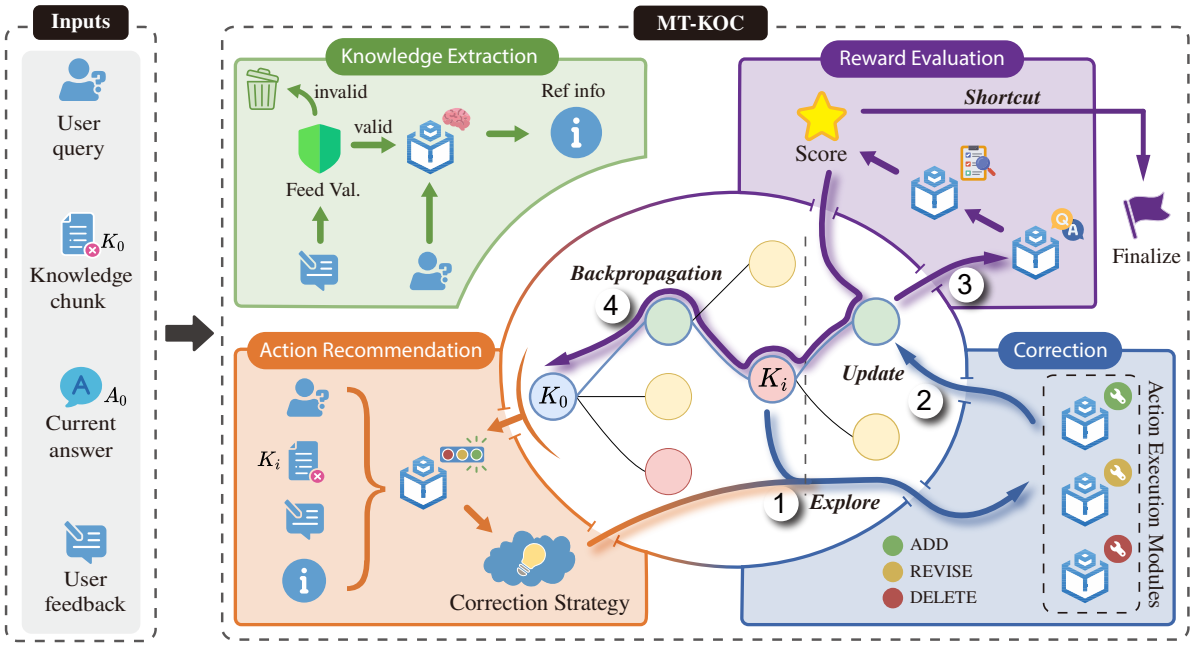


Figure 2: Overview of the MT-KOC framework. The process iteratively explores a search space of possible edits to find an optimal correction path for the Knowledge Chunk, guided by user feedback and a reward-driven mechanism.

the plausibility check, it guides the generation of I; otherwise, the agent synthesizes I based solely on the query (Q). Subsequently, the Action Recommendation Agent generates the initial set of candidate corrections from the root node.

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

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

$$C_{i+1} = \arg\max_{C \in \mathcal{C}(K_i)} \left(\frac{R(C)}{N(C)} + c\sqrt{\frac{\ln N(K_i)}{N(C)}} \right) \quad (5)$$

Here, $\arg\max_{C \in \mathcal{C}(K_i)}$ selects the action C_{i+1} from the candidate set $\mathcal{C}(K_i)$ that maximizes the UCT score. The UCT score balances exploitation and exploration: $\frac{R(C)}{N(C)}$ represents the average reward of action C, encouraging the selection of actions with historically high rewards, while $c\sqrt{\frac{\ln N(K_i)}{N(C)}}$ promotes exploration of less-visited actions, where $R(C)$ is the total reward accumulated for action C, $N(C)$ is the number of times action C has been selected, $N(K_i)$ is the number of visits to the parent node K_i, and c is an exploration constant.

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

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

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

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

Shortcut Mechanism (Pruning): To enhance search efficiency, we incorporate a pruning strategy. If at any point during the iterative search the Reward Evaluation Agent outputs a maximum score (e.g., 10.0), the search is immediately terminated. This early-stopping mechanism effectively prunes unnecessary future explorations, significantly reducing computational cost while ensuring the retrieval of an optimal solution. The Knowledge Chunk corresponding to this high-reward path is directly returned as K^*.

Default Selection: If the Shortcut is not triggered within the allocated search epochs, the system determines the optimal correction sequence $\mathbf{C}^* = (C_0^*, C_1^*, ..., C_L^*)$. This sequence is constructed by starting from the root node ($K_0^* = K_0$) and iteratively identifying the best transition. Specifically, at each step i, the optimal correction action C_{i+1}^* is selected from the candidate set $\mathcal{C}(K_i^*)$ by maximizing the average reward:

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

Here, $\arg\max_{C \in \mathcal{C}(K_i^*)}$ identifies the action C_{i+1}^* with the high-

Method	ConvCorrect-SQ			ConvCorrect-NB			ConvCorrect-HP			Avg
	DS-32B	GPT-5.1	Gemini-3	DS-32B	GPT-5.1	Gemini-3	DS-32B	GPT-5.1	Gemini-3	
Lower Bound										
Perturbed	13.2±0.2	13.0±0.1	15.1±0.2	29.5±0.3	31.6±0.2	33.2±0.2	16.9±0.1	18.7±0.2	23.7±0.3	21.7±0.2
Methods										
Re-Ex	54.1±0.3	55.5±0.2	57.0±0.2	35.8±0.1	38.1±0.3	39.5±0.2	52.3±0.3	54.8±0.2	57.1±0.2	49.4±0.2
CRAG	51.6±0.2	54.2±0.3	57.8±0.2	37.1±0.2	39.4±0.2	40.8±0.1	51.5±0.3	54.0±0.2	55.3±0.2	49.1±0.2
RAE	60.8±0.2	63.1±0.1	64.7±0.3	37.8±0.1	40.1±0.2	41.5±0.2	59.5±0.3	61.8±0.2	63.4±0.2	54.7±0.2
Astute	59.5±0.3	61.8±0.2	63.4±0.2	45.3±0.2	47.1±0.3	49.5±0.2	55.8±0.2	58.1±0.1	59.8±0.2	55.6±0.2
RARR	53.4±0.2	55.8±0.1	57.3±0.2	36.1±0.3	38.4±0.2	39.8±0.2	51.8±0.2	54.3±0.2	55.6±0.1	49.2±0.2
STACKFEED	65.3±0.2	67.8±0.2	67.3±0.3	44.8±0.2	47.3±0.2	48.8±0.3	57.8±0.1	60.3±0.2	61.8±0.2	57.9±0.2
MT-KOC	**67.3**±0.1	**70.2**±0.1	**73.5**±0.2	**47.8**±0.2	**51.5**±0.1	**54.2**±0.2	**65.5**±0.2	**69.8**±0.1	**72.0**±0.2	**63.5**±0.1
Upper Bound										
Oracle	82.8±0.1	85.6±0.0	87.0±0.1	51.6±0.1	53.4±0.1	58.2±0.1	77.6±0.1	80.3±0.1	82.0±0.1	73.2±0.1

Table 3: F1-score comparison on the three **ConvCorrect** benchmark splits. The best score in each column is **bolded**, and the second-best is underlined. DS-32B: DeepSeek-R1-Distill-Qwen-32B, Gemini-3: Gemini-3-Pro-Preview.

est average reward, ensuring a greedy selection of the best-performing actions to construct the optimal sequence. The final corrected Knowledge Chunk (K^*) is then obtained by applying this sequence $\mathbf{C}^*$ to K_0.

5 Experiments

5.1 Experimental Setup

Datasets. We evaluate our method on **ConvCorrect**, the benchmark we developed for interactive knowledge correction, consisting of three splits: ConvCorrect-SQ, ConvCorrect-NB, and ConvCorrect-HP (Section 3). 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. We compare MT-KOC against representative bounds and baselines:

- *Bounds:* We establish a performance floor with a *Lower Bound (Perturbed)*, which evaluates the answer generated from the erroneous chunk (K_0) without correction, and a ceiling with an *Upper Bound (Oracle)*, which uses the ground-truth knowledge chunk (K_{oracle}).
- *Baselines:* We compare against a comprehensive set of SOTA baselines, including **Re-Ex** (Kim et al., 2024), **CRAG** (Yan et al., 2024), **RAE** (Shi et al., 2024), **Astute** (Wang et al., 2025), **RARR** (Gao et al., 2023a), and **STACKFEED** (Gupta et al., 2024).

Evaluation Metrics. We evaluate performance from two perspectives:

- *Downstream Task Performance:* This metric evaluates the quality of the final answer. We report the token-level F1-score measuring the overlap between the generated answer (A^*) and the ground-truth answer (A_{gold}) provided in the dataset.
- *Knowledge Correction Quality:* This metric assesses the quality of the corrected knowledge chunk. We use ROUGE-L (Lin, 2004) to measure the similarity between the corrected chunk (K^*) and the ground-truth chunk (K_{oracle}) for accuracy, and between the original perturbed chunk (K_0) for edit minimality.

Implementation Details. To ensure reproducibility and fair comparison, we detail our experimental configurations below.

Base LLMs and Environment. To ensure robustness and reproducibility, all experiments were conducted across three independent runs with distinct random seeds, and we reported the average scores. Standard deviations across runs were consistently low (average ≤ 0.3) for the downstream task performance, indicating stable performance. Our experiments utilized three distinct LLMs: the open-weights DeepSeek-R1-Distill-Qwen-32B, and two proprietary LLMs, GPT-5.1 and Gemini-3-Pro-Preview. Inference for the open-weights LLM was performed on 8 NVIDIA H100 GPUs via vLLM framework, while proprietary LLMs were accessed through their official APIs. To ensure fairness, all methods in Table 3 shared identical backbones and input components (Q, K_0, F), with F integrated into respective prompts.

MT-KOC Configuration. For our proposed framework, we set the number of search epochs to 6. The exploration constant c in the UCT-based selection metric (Eq. 5) is set to 1.5. These values were selected based on a sensitivity analysis detailed in Appendix F. The Shortcut mechanism is triggered immediately when an evaluation score S_i from the Reward Evaluation Agent reaches the maximum value of 10. For the feedback validation step in the Knowledge Extraction Agent, we set the feedback confidence score threshold to 5.

5.2 Results and Analysis

Main Performance Comparison. Table 3 presents the F1-scores. MT-KOC achieves the highest performance across all three benchmarks (ConvCorrect-SQ, ConvCorrect-NB, and ConvCorrect-HP) and all base

Method	ConvCorrect-SQ		ConvCorrect-NB		ConvCorrect-HP	
	vs. Perturbed	vs. Oracle	vs. Perturbed	vs. Oracle	vs. Perturbed	vs. Oracle
Re-Ex	65.8±2.1	64.5±1.9	59.8±2.5	57.4±2.2	61.8±2.8	49.4±2.6
CRAG	62.1±1.8	60.8±2.0	70.6±2.4	52.1±2.3	59.5±2.5	47.8±2.7
RAE	63.8±1.7	62.3±1.8	60.5±2.1	58.8±2.0	62.5±2.4	50.8±2.5
Astute	65.1±1.9	63.5±1.8	59.1±2.3	56.8±2.1	63.3±2.6	52.1±2.4
RARR	60.8±2.4	59.5±2.2	54.2±2.8	50.5±2.5	59.1±3.1	46.3±2.9
STACKFEED	59.4±2.0	69.1±1.9	56.7±2.6	64.4±2.4	68.7±2.9	57.9±2.7
MT-KOC	**77.8**±1.2	**76.0**±1.1	**79.4**±1.5	**68.8**±1.4	**76.6**±1.6	**63.7**±1.8

Table 4: ROUGE-L similarity scores on the ConvCorrect-SQ, ConvCorrect-NB, and ConvCorrect-HP test sets. Results are averaged across all three base LLMs.

LLMs, consistently outperforming all baselines. In general, our method demonstrates superior effectiveness and robustness across different datasets and model scales.

Despite MT-KOC's competitive performance, a 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. These cases force the framework to fall back on the base LLM's own parametric knowledge (Lewis et al., 2020). This indicates that the primary performance bottleneck lies not with the MT-KOC framework itself, but with the knowledge breadth and reasoning depth of the underlying LLM, a limitation that could be addressed by integrating external knowledge retrieval capabilities or using LLMs with more extensive parametric knowledge.

Time Cost Analysis. As shown in Table 5, MT-KOC usually ends around four epochs, achieving an average latency of **20.3s**. This is due to the **85.4%** shortcut rate, which can largely reduce the time cost of the multi-agent system. This confirms that our method decreases the error correction delay from hours/days to seconds, which meets the requirements of the KOC task. Detailed analysis is in Appendix G.

Benchmark Split	Epochs	Shortcut (%)	Latency (s)
ConvCorrect-SQ	3.47±0.12	90.7±3.5	13.4±0.8
ConvCorrect-NB	3.77±0.15	86.0±4.2	21.8±1.1
ConvCorrect-HP	4.02±0.18	79.4±3.8	25.7±0.9
Average	3.75±0.15	85.4±3.9	20.3±1.0

Table 5: Efficiency metrics of MT-KOC using GPT-5.1. Shortcut Rate is the percentage of samples terminating before the maximum 6 epochs.

Knowledge Correction Quality Analysis. To evaluate the quality of the knowledge correction, we report the ROUGE-L scores averaged across all three base LLMs in Table 4. Firstly, MT-KOC achieves the highest ROUGE-L against (K_{oracle}), demonstrating that the corrected chunk (K^*) is semantically closest to the ground-truth correct version. Secondly, it also achieves the highest score against the original perturbed chunk (K_0), indicating superior edit minimality by preserving the maximum amount of correct and unchanged information from K_0 while effectively correcting only the erroneous parts.

5.3 Ablation Study

To validate the contribution of the key components of our methods, we conducted ablation studies across all three datasets. As shown in Table 6, removing **Multi-Path Search** reduces the method to a single greedy path and degrades performance across all datasets, highlighting the importance of exploring multiple correction trajectories to escape local optima. Disabling the **Shortcut Mechanism** also lowers scores, suggesting it prevents unnecessary edits once an optimal solution is reached. Finally, removing **Feedback Validation** causes a significant drop; without it, the system accepts abnormal feedback in our benchmark and injects errors into the knowledge base.

Method	SQ	NB	HP
MT-KOC	70.2	51.5	69.8
w/o M-P Search	67.8 (↓ 2.4)	48.2 (↓ 3.3)	65.5 (↓ 4.3)
w/o Shortcut	67.9 (↓ 2.3)	50.4 (↓ 1.1)	68.6 (↓ 1.2)
w/o Feedback Val.	68.4 (↓ 1.8)	49.3 (↓ 2.2)	67.1 (↓ 2.7)

Table 6: Ablation study of MT-KOC. Results are reported in F1-score using **GPT-5.1** as the base LLM.

6 Conclusion

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

Limitations

We discuss two limitations of our work in this section:

(1) Although ConvCorrect incorporates three diverse datasets, they are subject to limitations in both domain and scale. First, the data is primarily derived from general-domain sources, which may not fully generalize to highly specialized fields like legal or medical documentation. Second, with the largest split containing approximately 130,000 samples, the scale is relatively modest compared to some massive-scale RAG benchmarks. Future expansions to broader domains and larger scales would further validate the framework.

(2) We primarily focus on the immediate accuracy of corrections. While effective within the current conversational session, the long-term persistence and potential side effects of these updates on the knowledge base remain to be explored.

Ethical Considerations

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

We acknowledge the use of LLM for grammar checking and linguistic polishing of the manuscript. The final content was thoroughly reviewed and approved by all authors, we maintain full responsibility for the work's integrity. The study has been reviewed and approved by the Institutional Review Board (IRB) of the authors' institution.

References

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

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and 2 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901.

Anthony Chen, Panupong Pasupat, Sameer Singh, Hongrae Lee, and Kelvin Guu. 2023. Purr: Efficiently editing language model hallucinations by denoising language model corruptions. *arXiv preprint arXiv:2305.14908*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, and 1 others. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

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

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

Guanting Dong, Jiajie Jin, Xiaoxi Li, Yutao Zhu, Zhicheng Dou, and Ji-Rong Wen. 2025. RAG-critic: Leveraging automated critic-guided agentic workflow for retrieval augmented generation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 3551–3578.

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

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

Naman Gupta, Shashank Kirtania, Priyanshu Gupta, Krishna K. Mehra, Sumit Gulwani, Arun Iyer, Suresh Parthasarathy, Arjun Radhakrishna, Sriram K. Rajamani, and Gustavo Soares. 2024. Stackfeed: Structured textual actor-critic knowledge base editing with feedback. *arXiv preprint arXiv:2410.10584*.

Hangfeng He, Hongming Zhang, and Dan Roth. 2023. Rethinking with retrieval: Faithful large language model inference. *CoRR*.

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

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

Juyeon Kim, Jeongeun Lee, Yoonho Chang, Chanyeol Choi, Junseong Kim, and Jy-yong Sohn. 2024. Re-ex: Revising after explanation reduces the factual errors in LLM responses. *CoRR*.

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

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

Changmao Li and Jeffrey Flanigan. 2025. RAC: Efficient LLM factuality correction with retrieval augmentation. In *Findings of the Association for Computational Linguistics: EMNLP*, pages 25145–25159. Association for Computational Linguistics.

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

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81. Association for Computational Linguistics.

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

Ahmed Masry, Mohammed Saidul Islam, Mahir Ahmed, Aayush Bajaj, Firoz Kabir, Aaryaman Kartha, Md Tahmid Rahman Laskar, Mizanur Rahman, Shadikur Rahman, Mehrad Shahmohammadi, and 1 others. 2025. Chartqapro: A more diverse and challenging benchmark for chart question answering. *arXiv preprint arXiv:2504.05506*.

Zachary McGraw. 2025. What triggered the google cloud outage in june 2025? https://www.vcsolutions.com/blog/google-cloud-outage-causes-behind-\the-june-2025-incident/. Last accessed: 2025-09-15.

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

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

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

Zile Qiao, Wei Ye, Yong Jiang, Tong Mo, Pengjun Xie, Weiping Li, Fei Huang, and Shikun Zhang. 2025. Supportiveness-based knowledge rewriting for retrieval-augmented language modeling. In *Findings of the Association for Computational Linguistics*, pages 2728–2740.

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

Q Shi and 1 others. 2024. Retrieval-enhanced knowledge editing in language models for multi-hop question answering. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM)*, pages 2056–2066.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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

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

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

Yuan-Quan Wang, Ying-Ying Zhang, and Jia-Lei Liu. 2023. Expectation identity of the hypergeometric

distribution and its application in the calculations of high-order origin moments. *Communications in Statistics-Theory and Methods*, pages 6018–6036.

Di Wu, Jia-Chen Gu, Fan Yin, Nanyun Peng, and Kai-Wei Chang. 2024. Synchronous faithfulness monitoring for trustworthy retrieval-augmented generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 9390–9406.

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

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

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

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

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

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

Yunxiang Zhang, Muhammad Khalifa, Lajanugen Logeswaran, Moontae Lee, Honglak Lee, and Lu Wang. 2023b. Merging generated and retrieved knowledge for open-domain QA. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 4710–4728.

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

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

Appendix

A Sample with Each Error Type

This section provides concrete examples for each of the five error types defined in our benchmark. We use the same user query and oracle knowledge context across all examples to clearly illustrate how different perturbations affect the knowledge chunk and the subsequent LLM-generated answer.

A.1 Shared Context for All Examples

The following User Query and Oracle Knowledge serve as the baseline ground truth.

User Query (Q)

Please describe the key identifying features of the Death Cap mushroom.

Oracle Knowledge Chunk (K_{oracle})

...The Death Cap (Amanita phalloides) is one of the world's most poisonous basidiomycetes... Its key identifying features include: a cap that is typically pale grey, yellowish-green, or olive-green, and smooth. The gills are white and free (i.e., not attached directly to the stipe). The most important feature is that the base of its stipe has a distinct, cup-like volva, and the upper part of the stipe usually has an annulus (ring)...

Correct Answer (A_{oracle})

The key features of the Death Cap are: 1) A cap color of pale grey, yellowish-green, or olive-green; 2) Gills that are white and free; and 3) A distinct, cup-like volva structure at the base of the stipe.

A.2 Type 1: Fully Incorrect

The knowledge chunk contains completely fabricated or contradictory information.

Erroneous Knowledge Chunk (K)

...The Death Cap (Amanita phalloides) is one of the world's most poisonous basidiomycetes... Its key identifying features include: a cap that is typically bright red with white warts. The gills are yellow and decurrent (i.e., running down the stipe). The base of its stipe lacks any special cup-like structure, simply tapering at the base, and 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 that is bright red with white warts; 2) Gills that are yellow and decurrent; and 3) No cup-like structure at the base of its stipe.

A.3 Type 2: Partially Incorrect

The chunk contains a mix of correct facts and specific hallucinations (e.g., incorrect color).

Erroneous Knowledge Chunk (K)

...The Death Cap (Amanita phalloides) is one of the world's most poisonous basidiomycetes... Its key identifying features include: a cap that is typically pure white or dark brown, and smooth. The gills are white and free (i.e., not attached directly to the stipe). The most important feature is that the base of its stipe has a distinct, cup-like volva...

Generated Answer from K (A)

The key features of the Death Cap include: 1) A cap color of pure white or dark brown; 2) Gills that are white and free; and 3) A cup-like volva structure at the base of the stipe.

A.4 Type 3: Fully Missing

The chunk discusses the general topic but omits the specific evidence required to answer the query.

Erroneous Knowledge Chunk (K)

...The Death Cap (Amanita phalloides) is one of the world's most poisonous basidiomycetes and is responsible for the vast majority of fatal mushroom poisonings worldwide. Its primary toxin is α-amanitin, a heat-stable cyclopeptide that cannot be destroyed by cooking. Symptoms of poisoning typically do not appear until 6 to 12 hours after consumption...

Generated Answer from K (A)

The provided text does not describe the physical identifying features of the Death Cap. It only discusses its extreme toxicity, the toxin α-amanitin, and the delayed symptoms of poisoning.

A.5 Type 4: Partially Missing

The chunk retains some correct details but removes other key information required for a comprehensive answer.

Erroneous Knowledge Chunk (K)

...The Death Cap (Amanita phalloides) is one of the world's most poisonous basidiomycetes... Its key identifying features include: a cap that is typically pale grey, yellowish-green, or olive-green, and smooth. The gills are white and free (i.e., not attached directly to 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.

A.6 Type 5: Mixed

The chunk contains a combination of correct, incorrect, and missing information.

Erroneous Knowledge Chunk (K)

...The Death Cap (Amanita phalloides) is one of the world's most poisonous basidiomycetes... Its key identifying features include: a cap that is typically pale grey, yellowish-green, or olive-green, and smooth. The gills are bright pink and attached. 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 bright pink and attached.

B Instructions for Preliminary User Study

The following text presents the complete instructions provided to the participants in our preliminary user study.

Instructions for Participants

Thank you for your help! We are studying how people naturally react when an AI makes a mistake.

The Scenario

In this task, we are simulating a very common situation:

- You ask an AI assistant a question.
- The AI gives you an answer that is flawed, incomplete, or not quite right.

Your Role & Task

You are the user who asked the original question. Your task is simple:
In the text box for each item, type *exactly* what you would say or write next in response to the AI's flawed answer.

Your Goal: Be 100% Natural

- Give your honest, spontaneous reaction. Don't try to be extra polite or extra helpful unless that's what you would actually do.
- There are no rules. Your response could be a correction, a new question, a simple word, or even a sign of frustration. We want it all!
- No right or wrong answers. We are purely interested in what a real person would do in this situation.

C Prompts in Benchmark Construction

This section details the prompts in our benchmark construction: context perturbation (Figure 3), perturbation evaluation (Figure 4), and feedback generation (Figure 5).

D Prompts in Multi-Agent System

This section provides the detailed prompts used for each agent in the MT-KOC framework. These prompts, cor-

Feedback Style	Error Type					Total
	Fully Inc.	Partially Inc.	Fully Miss.	Partially Miss.	Mixed	
Direct correction	352	458	-	-	-	**810**
Error indication	298	415	-	-	-	**713**
Error localization	-	312	-	-	-	**312**
Direct completion	-	-	19	118	-	**137**
Missing indication	-	-	21	105	-	**126**
Direct correction and completion	-	-	-	-	25	**25**
Error and missing indication	-	-	-	-	23	**23**
Abnormal Feedback	76	142	5	24	7	**254**
Total Samples	**726**	**1,327**	**45**	**247**	**55**	**2,400**

Table 7: Detailed distribution of 2,400 user feedback responses collected in the preliminary study. The counts represent the number of participants who chose a specific feedback style for a given error type.

responding to the Knowledge Extraction Agent (Figure 6), Action Recommendation Agent (Figure 7), Correction Agent modules (Figure 8), and Reward Evaluation Agent modules (Figure 9), are presented with maximum fidelity to the implementation code to ensure reproducibility.

E Human Evaluation of Reward Evaluation Agent

To assess the reliability of our Reward Evaluation Agent and its alignment with human preferences, we conducted a human correlation study. We drew a stratified random sample covering 5% of instances from each of the three benchmark splits. The sampled instances were evaluated by an expert annotator following the exact same 0–10 scoring rubric as the Reward Agent.

Agreement Metric	Value
Mean Absolute Error (MAE)	1.14
Exact Agreement (diff=0)	31%
Agreement within ± 1 Point	69%
Agreement within ± 2 Points	88%

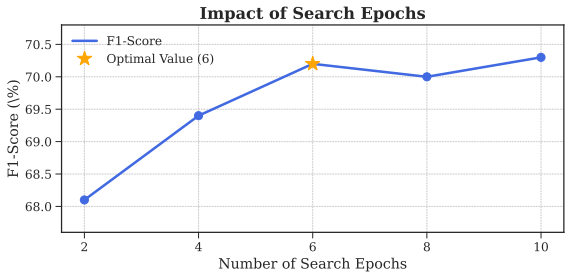
Table 9: Agreement analysis between Reward Evaluation Agent scores and human ratings.

We measured the agreement between the LLM-generated scores and the human ratings using intuitive error and tolerance metrics, as presented in Table 9, the results indicate a satisfactory alignment with human judgments. These metrics suggest that our automated Reward Evaluation Agent can serve as a reliable and scalable proxy for human evaluation within the MT-KOC.

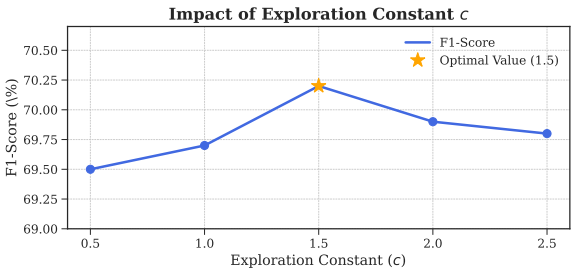
F Hyperparameter Sensitivity Analysis

To select the hyperparameters for MT-KOC, we conducted a sensitivity analysis on the ConvCorrect-SQ dataset using GPT-5.1 as the base LLM. We varied the number of search epochs and the exploration constant c and measured the impact on F1-score.

The results are shown in Figure 10. For the number of search epochs (Figure 10a), we observed that performance peaks at 10 epochs (70.3 F1). However, the performance at **6 epochs** (70.2 F1) is highly competitive, with only a marginal 0.1-point difference. Given that increasing the epochs from 6 to 10 significantly raises the computational cost, we selected **epochs=6** as our final setting to achieve a better balance between performance and efficiency. For the exploration constant (Figure 10b), performance is optimal when c is set to **1.5**. Based on this analysis, we used epochs=6 and c=1.5 for all our main experiments.



(a) Impact of Search Epochs (c fixed at 1.5)



(b) Impact of Exploration Constant c (epochs fixed at 6)

Figure 10: Hyperparameter sensitivity analysis for MT-KOC on the ConvCorrect-SQ dataset.

Statistic	ConvCorrect-SQ	ConvCorrect-NB	ConvCorrect-HP
Total Samples	130,444	93,707	49,130
Error Type Distribution			
Fully incorrect	77.6%	29.6%	62.4%
Partially incorrect	11.5%	57.3%	15.8%
Fully missing	0.2%	0.3%	7.6%
Partially missing	10.3%	10.1%	14.0%
Mixed	0.4%	2.7%	0.2%
Feedback Style Distribution			
Direct correction	40.8%	33.5%	34.9%
Error indication	40.7%	32.6%	34.9%
Error localization	3.7%	18.5%	5.1%
Direct completion	5.0%	4.9%	10.3%
Missing indication	5.0%	4.9%	10.3%
Direct correction and completion	0.1%	0.9%	0.1%
Error and missing indication	0.1%	0.9%	0.1%
Abnormal Feedback	4.6%	3.8%	4.5%

Table 8: Detailed statistics of the ConvCorrect Benchmark, showing the scale and the distribution of Error Types and Feedback Styles across the three dataset splits: ConvCorrect-SQ, ConvCorrect-NB, and ConvCorrect-HP.

G Detailed Efficiency Analysis

Here, we examine the search depth of MT-KOC to understand its shortcut mechanism. Table 10 details the exit epoch distribution, with average trends illustrated in Figure 11. We observe a search distribution centered on 3–4 epochs. While most samples are resolved efficiently (achieving a 90.7% shortcut rate on ConvCorrect-SQ), the tail of the distribution reveals that complex scenarios, such as those in ConvCorrect-HP, trigger deeper searches. This confirms that MT-KOC effectively balances low latency for simple inputs with the thoroughness required for harder corrections.

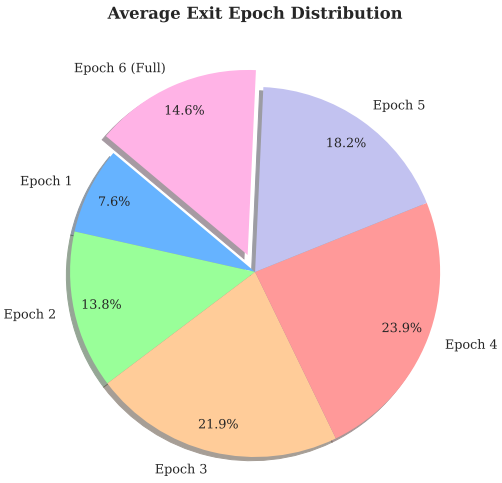


Figure 11: Average distribution of exit epochs for the dynamic action search process across all benchmarks.

Exit Epoch	SQ (%)	NB (%)	HP (%)	Avg. (%)
Epoch 1	9.6±1.2	6.9±0.9	6.2±0.8	7.6±0.6
Epoch 2	16.4±1.5	13.7±1.3	11.2±1.1	13.8±0.7
Epoch 3	25.0±2.1	21.9±1.8	18.8±1.7	21.9±1.1
Epoch 4	24.3±1.9	24.9±2.0	22.6±1.9	23.9±1.1
Epoch 5	15.3±1.4	18.6±1.6	20.6±1.8	18.2±0.9
Epoch 6 (Full)	9.4±1.1	14.0±1.4	20.6±2.2	14.7±0.9

Table 10: Detailed distribution of exit epochs across the three benchmark splits (**SQ**: ConvCorrect-SQ, **NB**: ConvCorrect-NB, **HP**: ConvCorrect-HP).

H Reproducibility Statement

To ensure the reproducibility of our results, we provide comprehensive details on our benchmark, methodology, and experimental setup. The construction process of the ConvCorrect benchmark is outlined in Section 3. The architecture and dynamic action search algorithm of our MT-KOC method are described in Section 4. Full implementation details are provided in Section 5.1. The benchmark data will be made publicly available under the Apache 2.0 License upon publication.

Context Perturbation

You are a text perturbation agent cascading to simulate critical errors in a knowledge base, ensuring the core semantic content related to the correct answer is completely disrupted to prevent its generation. Your goal is to aggressively alter answer-related information to make the correct answer impossible to derive while preserving all content unrelated to the question unchanged.
Guidelines:

1. Preserve *all* content unrelated to the question (e.g., sentences, formatting, titles, introductions, metadata, comments) exactly as in the original, allowing only minor differences in spacing or punctuation.
2. Identify sentences or phrases that directly contain or imply the correct answer (e.g., exact answer text, synonyms, or related facts).
3. Apply the specified perturbation type to those sentences or phrases to drastically alter their semantic meaning:
   - `"replace"`: Substitute with strongly contradictory or completely unrelated information that fundamentally changes the meaning (e.g., opposite facts, different entities, or unrelated concepts, ensuring no trace of the original answer remains).
   - `"add"`: Append strongly contradictory or completely unrelated details immediately after the answer-related content, explicitly negating or overshadowing the original information to make the correct answer impossible to derive.
   - `"delete"`: Completely remove all sentences or phrases that reference the answer, including synonyms or related facts, with no residual presence.
4. Ensure the perturbed context does not contain the correct answer or its synonyms in answer-related sentences.
5. Make aggressive changes to answer-related content to ensure the correct answer cannot be inferred, prioritizing maximal disruption over minimal changes.
6. Output ONLY the perturbed context as plain text, preserving all content unrelated to the question.

[User Input Template]
Perturbation Type: [Perturbation Type]
Original Context: [Original Context]
Question: [Question]
Correct Answer: [Answer]

Instructions:

1. Preserve *all* content unrelated to the question exactly as in the original, allowing only minor differences in spacing or punctuation.
2. Identify sentences or phrases that directly contain or implying the correct answer: “[Answer]”.
3. Apply the perturbation type ([Perturbation Type]) to those parts to drastically alter their semantic meaning.
4. Ensure the perturbed context does not contain the correct answer or its synonyms in answer-related sentences.
5. Make aggressive changes to answer-related content to prevent deriving the correct answer.
6. Output ONLY the perturbed context as plain text, preserving all content unrelated to the question.

Output:

Figure 3: Prompt for the Context Perturbation Agent.

Perturbation Evaluation

You are an evaluation agent tasked with assessing whether a perturbed context drastically alters the core semantic content related to the correct answer while preserving all content unrelated to the question. Your goal is to verify if the perturbation prevents generating the correct answer. Output a single-line JSON response with "is_valid" (true/false) and "reason" (explanation).

Requirements to check:

1. All content unrelated to the question (e.g., sentences, structure, titles, introductions, metadata) must be preserved exactly as in the original, allowing only minor differences in spacing or punctuation.
2. Answer-related content (sentences/phrases containing or implying the correct answer) must be drastically altered to prevent generating the correct answer according to the perturbation type (`"replace"`, `"add"`, or `"delete"`).
3. The output must avoid artificial terms like "ERROR", "WRONG", or "[placeholder]".

Output format (single-line JSON):

```
{"is_valid": true/false, "reason": "Explanation of why the perturbation is valid or invalid"}
```

[User Input Template]

Perturbation Type: [Perturbation Type]
Original Context: [Original Context]
Perturbed Context: [Perturbed Context]
Question: [Question]
Correct Answer: [Answer]

Instructions:

1. Evaluate whether the perturbed context meets the perturbation requirements.
2. Check if:
   - Answer-related content (containing or implying "[Answer]") is drastically altered per the perturbation type to prevent deriving the correct answer.
   - All content unrelated to the question is preserved exactly, allowing minor differences in spacing or punctuation.
3. Output a single-line JSON object with "is_valid" and "reason".

Output:

Figure 4: Prompt for the Perturbation Evaluation Agent.

Feedback Generation (Semi-automated)

You are a friendly and knowledgeable AI assistant tasked with generating natural, conversational, and varied feedback for the current model response based on a question, the correct answer, and the identified error type. Follow these guidelines:

1. Generate feedback that matches the specified feedback type and error type.
2. Use the following feedback styles and their definitions:
   - **Direct correction**: Highlight the specific error in the current model response and provide the correct answer with a brief, clear explanation.
   - **Error indication**: Make a general comment indicating the error type without detailed correction.
   - **Error localization** / **Missing indication**: Identify the specific error or location in the current model response without giving the correct answer or extra details.
   - **Direct completion**: Share the correct answer and relevant details to fill in completely missing information.
   - **Direct completion (Supplemental)**: Add the missing core information to the current model response without changing what was provided.
   - **Direct correction and completion**: Correct the wrong core information in the current model response and add any missing details.
   - **Direct correction (Partial)**: Fix only the incorrect part of the current model response without addressing missing information.
   - **Abnormal feedback**: This simulates a non-cooperative user. Generate feedback that is one of the following:
     – (a) **Malicious/Rude**: Use mildly aggressive or insulting language.
     – (b) **Intentionally Misleading**: Confidently provide a WRONG fact as a correction.
     – (c) **Irrelevant**: Say something completely unrelated to the conversation.
     For Abnormal feedback, ignore the Correct Answer and Error Type constraints.
3. Keep feedback concise, natural, and conversational.
4. **Crucially, mimic the style, tone, and length of the provided human example below.**
5. Output ONLY the feedback message as plain text. Do NOT include JSON, extra explanations, or other text.

[Human Example Section]
The following example is dynamically sampled from our user study database based on the target Feedback Style:
Real Human Example:
Question: [Example Question]
Model Response: [Example Model Response]
User Feedback: [Example User Feedback]

[User Input Template]
Question: [Question]
Correct Answer: [Correct Answer]
Current Model Response: [Model Response]
Error Type: [Error Type]
Generate feedback for the feedback style: [Feedback Style]. Output only the feedback message as plain text.

Figure 5: Prompt for the Feedback Generation Agent using a semi-automated approach with human examples.

Knowledge Extraction Agent

You are a highly efficient Knowledge Agent with two primary functions, executed in a single step.
Function 1: Feedback Plausibility Check
Evaluate the [User Feedback] for its plausibility against your general world knowledge. Assign a confidence score from 0 to 10.
- **Score 9-10**: Highly plausible/Correct (e.g., "The capital is Paris").
- **Score 5-8**: Plausible but niche/debatable.
- **Score 0-4**: Highly implausible, nonsensical, or malicious (e.g., "The Earth is flat").

Function 2: Adaptive Fact Extraction
Based on your score, synthesize a concise "ground truth" snippet (Reference Information).
- **If Score >= 5**: Trust the feedback. Extract the most relevant fact based on BOTH the [Query] AND the [Feedback].
- **If Score < 5**: DISTRUST the feedback. IGNORE the feedback content. Extract the correct fact based ONLY on the [Query] and your internal knowledge.

Input Format:
[Query]: The user's original question.
[Feedback]: The user's feedback.
Output Format:
You MUST respond with ONLY a valid JSON object with the following keys:
- `feedback_validation`: An object containing `confidence_score` (int) and `justification` (str).
- `reference_info`: A string containing the extracted fact.

Example (Valid Feedback):
[Query]: "Who directed Titanic?"
[Feedback]: "It was James Cameron."
[Output]:

```
{
   "feedback_validation": {
     "confidence_score": 10,
     "justification": "Feedback is factual."
   },
   "reference_info": "The movie Titanic was directed by James Cameron."
}
```

Example (Invalid Feedback):
[Query]: "Who directed Titanic?"
[Feedback]: "It was directed by a potato."
[Output]:

```
{
   "feedback_validation": {
     "confidence_score": 0,
     "justification": "Feedback is nonsensical."
   },
   "reference_info": "The movie Titanic was directed by James Cameron."
}
```

Figure 6: Prompt for the Knowledge Extraction Agent.

Action Recommendation Agent

You are a brilliant AI Recommendation Generator. Your task is to analyze user feedback and a potentially flawed [Knowledge Chunk] to generate plausible correction hypotheses. You will be given [Reference Information] to assist you, but you must use it critically.

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.
   - **Vague Feedback**: If the [Feedback] is vague (e.g., "That's wrong"), 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.
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.

Figure 7: Prompt for the Action Recommendation Agent.

Correction Agent: ADD Module

You are a precise AI text editor. Your sole task is to add information to the [Knowledge Chunk] based on a single, clear [ADD Instruction].
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.

Correction Agent: REVISE Module

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

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.

Correction Agent: DELETE Module

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

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

Figure 8: Prompts for the three modules of the Correction Agent.

Reward Evaluation Agent: Answer Generation Module

You are a highly efficient assistant focused on extracting precise answers. Use the provided context to answer the user's question. Follow these strict rules:

1. Output ONLY the key entity, name, date, number, or phrase that answers the question. Do not use complete sentences.
2. Do not use articles (a, an, the) or introductory phrases (e.g., "The answer is", "It is") unless part of a proper name.
3. If the answer is not explicitly stated, use reasonable inference from the context to provide the best possible short answer.

Reward Evaluation Agent: Scoring Module

You are a strict and objective Answer Evaluator. Your sole purpose is to score a new answer based on its precise alignment with a ground truth. Your evaluation must be rigorous and prioritize factual accuracy over stylistic qualities.
Evaluation Protocol (Strict Priority Order):

1. **Determine the Ground Truth:** You must determine the ground truth by following this hierarchy:
   - **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.
   - **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**.
   - **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.
2. **Score based on Ground Truth Alignment (0–10):**
   - **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.
   - **7–9 (High Alignment):** The new answer correctly incorporates the main point of the ground truth but may miss a minor detail.
   - **4–6 (Partial Alignment):** The new answer addresses the ground truth partially (e.g., corrects one error but misses another).
   - **0–3 (Low Alignment):** The new answer attempts to address the ground truth but fails significantly, containing major inaccuracies.

Scoring Rules:

- **Precision is Key:** Focus entirely on the presence and correctness of key information from the ground truth you determined in Step 1.
- **Output ONLY the numerical score (0–10).**

Figure 9: Prompts for the two modules of the Reward Evaluation Agent.