Mutable Transcripts: Mitigating Context Pollution through Editable Conversation State

Dan Barry
School of Computer Science
University College Dublin
Dublin, Ireland
danbarry@duck.com

Andrew Hines
School of Computer Science
University College Dublin
Dublin, Ireland
andrew.hines@ucd.ie

Abstract

Contemporary large language model (LLM) chat systems treat conversation history as an immutable sequence of turns that defines the model's working context. However, user intent in real interactions is not static: it evolves through correction, refinement, and shifting constraints. This mismatch between dynamic intent and static transcripts can result in context pollution, where outdated or irrelevant information persists and continues to influence subsequent responses. We introduce mutable transcripts, a new interaction paradigm that enables users to revise prior turns through natural language edit requests, allowing the conversation history itself to be updated rather than appended. This reframes the transcript from a passive record into an editable representation of conversational state. We present a working prototype that integrates transcript-level revision into a standard chat interface and evaluate its feasibility through a controlled user study (n=17) and an illustrative transcript analysis of representative interaction scenarios. Participants significantly preferred mutable transcripts over standard chat across measures of clarity, confidence, and ease of use, with reduced intent to restart conversations. Transcript analysis of representative user study conversations shows that mutable transcripts can reduce conversation length and eliminate obsolete retained context. These findings provide initial evidence that user-driven revision of conversational history can improve interaction quality and help maintain a more current representation of user intent. The source code and prototype can be accessed at https://github.com/QxLabIreland/ReChat.

1 Introduction

Large language model (LLM) chat interfaces are now commonplace, shaping how people explore ideas [1], solve problems [2], and draft content [3]. The experience is typically a familiar chat layout: a scrolling conversation with alternating user and assistant messages, a prompt box at the bottom, and lightweight feedback such as typing indicators or message streaming. This interaction model has become the de facto standard for mainstream services like ChatGPT and similar assistants, setting user expectations around simplicity, speed, and conversational flow [4]. However, this model implicitly assumes that conversational history should be accumulated rather than revised, an assumption that introduces challenges as user intent evolves over time.

1.1 How Chat Context Works

At a high level, most chat systems are stateless between turns. The only "memory" is the conversation history itself, which is sent to the model each time the user asks a new question [5]. In practice this is often represented as a JSON transcript of alternating user and assistant messages that is passed back and forth on every request [6]. The model processes the full transcript to produce a contextually

relevant response, which means the payload to the LLM grows as the conversation length increases. Over time, the growing transcript becomes both a technical and UX constraint, and it sets the stage for the context management challenges discussed in the next section.

1.2 Context Pollution

This conversation history is the context for the entire thread, so any confusion or revision accumulates alongside the useful material. As conversations grow, they can become unwieldy, making it harder for the model to maintain coherence [7]. Users also correct themselves over time: a misunderstanding gets resolved, a constraint changes, or a creative direction shifts. If you are using an AI as a writing partner, for example, you might decide later that a character's motives are different than originally stated.

These changes are reasonable in the moment, but they contribute to what is known as context pollution. Context pollution is the accumulation of outdated, contradictory, or no-longer-relevant information in a conversation transcript that continues to influence the model's responses, even after the user's intent has changed [7, 8]. The accumulation of obsolete information persists because current systems operate on the basis of appending corrective turns instead of revising or replacing stale concepts. The model can then surface earlier, stale concepts in later turns, not because it is "forgetting," but because it is struggling to know what it should forget based on user feedback in prior turns.

Prior work on AI-assisted interaction has explored several approaches to managing long or evolving conversational context, including context window truncation [7], automatic summarization [9, 10], and retrieval-augmented memory [11], but these techniques treat context management as a system-driven optimization rather than a user-controlled activity. As a result, users lack direct mechanisms to align the retained conversational state with their current intent. Other interfaces move beyond linear chat by embedding language models in document editors, notebooks, or agent frameworks, where users can revise inputs directly or restructure intermediate artifacts [12]. However, these approaches typically abandon the conversational metaphor altogether. In contrast, mutable transcripts preserve the familiar chat interface while introducing explicit, user-driven revision of conversational history. Rather than compressing, hiding, or implicitly rewriting prior context, mutable transcripts surface transcript modification as a first-class interaction, allowing users to correct, prune, or reframe earlier turns in ways that existing chat systems do not support.

1.3 The Immutability Constraint

Current chat interfaces typically follow one of two user-driven edit patterns: (1) they allow only the most recent user message to be edited (Gemini and Duck.ai), or (2) they allow edits at any point in history but do not revise previous or subsequent messages, instead creating a conversation branch (ChatGPT and Claude) in which further corrective turns are appended. While branching supports exploratory workflows, it preserves prior context rather than revising it, requiring users to manage multiple divergent threads while outdated or incorrect information remains embedded in earlier turns. As a result, earlier turns are effectively treated as fixed, creating a history that users can read but not fully revise. This constraint is partly a product concern: editable transcripts could introduce new security risks, such as prompt-injection attacks that rewrite or disguise earlier context [13], and they complicate accountability because the model's behavior depends on an evolving history.

There is also a UX challenge in making mutable history feel trustworthy and understandable. Interfaces would need to communicate what changed, when it changed, and why the system's responses shifted. Despite these challenges, an open interaction question remains: how would users experience a conversational interface in which transcript history could be revised directly? In this work, we focus specifically on the interaction and usability implications of editable conversational state, rather than attempting to resolve the broader security or accountability challenges introduced by mutable history.

Our contributions are threefold:

1. We identify context pollution as a consequence of immutable transcript design in iterative human-AI chat interaction.
2. We propose mutable transcripts, which enable users to revise prior conversational state through natural language edit requests while preserving the chat interface metaphor.

3. We demonstrate the feasibility of this approach through a working prototype, a controlled user study, and an illustrative transcript analysis showing notable reductions in obsolete retained context in representative scenarios. The source code and prototype can be accessed at https://github.com/QxLabIreland/ReChat.

This work is best understood as a concept and feasibility study. Our goal is not to provide a definitive empirical comparison across all interaction paradigms, but to introduce and validate a new interaction model with strong early evidence of utility. As such, we prioritize demonstrating feasibility, usability, and structural effects, while leaving large-scale benchmarking and comparative evaluation to future work.

2 Mutable Transcripts

To address context pollution, we introduce *mutable transcripts* as an interaction paradigm for managing conversational context in AI chat systems. In this paradigm, the conversation transcript is treated not as an immutable log, but as an editable representation of conversational state that can be revised as user intent evolves.

Formally, a mutable transcript allows a user to issue a natural language edit request that operates over the full conversation history, producing a revised transcript in which both user and assistant messages are updated to reflect the requested change. This enables retroactive correction, constraint injection, and context pruning without requiring users to manually edit individual messages or restart the conversation. A mutable transcript can be viewed as a function $T' = f(T, e)$, where T is the original transcript and e is a natural language edit instruction.

This reframing shifts conversational interaction from an append-only model, where corrections accumulate as additional turns, to a revision-based model in which outdated or contradictory information can be removed or updated directly. As a result, the retained conversational state more closely reflects the user's current intent.

We explore the feasibility of this approach through a working prototype integrated with an LLM, along with case studies and a controlled user study.

3 System Design

We implement mutable transcripts within a standard chat interface augmented with a lightweight interaction mode for transcript revision. The system operates in two modes: a default append-only mode and an "Edit History" mode that enables transcript-level modification.

In default mode, the system behaves as a conventional chat interface, where each new user input is appended to the existing transcript and passed to the model to generate a response. In "Edit History" mode, the user instead issues a natural language edit request that operates over the entire conversation history.

Figure 1 illustrates the interaction flow. Upon activating "Edit History" mode, the interface provides visual cues (Figure 1A) to indicate that subsequent input will be interpreted as a revision request. The user then submits an edit instruction, after which the system rewrites the full transcript (Figure 1B) and replaces the displayed conversation with the updated version (Figure 1C).

Using natural language as the editing interface avoids introducing complex manual editing controls, allowing transcript revision to remain consistent with existing chat interaction patterns. The only additional interface element is the mode selector, making the transition from standard chat to mutable transcripts low-friction. Unlike direct editing of individual messages, natural-language revision allows users to specify changes spanning multiple turns while delegating the corresponding transcript restructuring to the system.

3.1 Architecture

The system implements transcript revision as a transformation over the full conversation history. Given an original transcript T and a natural language edit instruction e, the system produces a revised transcript $T' = f(T, e)$.

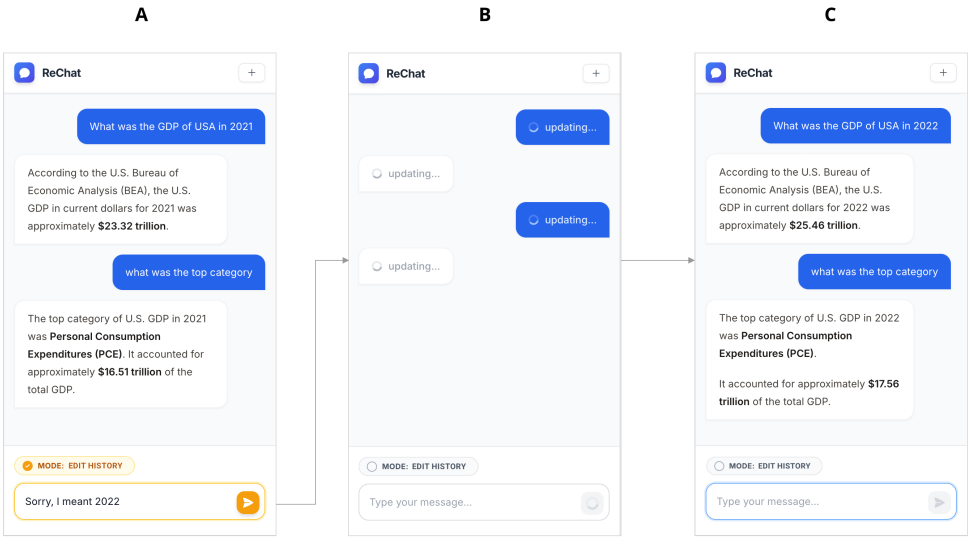


Figure 1: Interface design: (A) At any point in the conversation, the user can engage "Edit History" mode and enter a prompt to modify the chat history. (B) The UI refreshes and shows that previous conversational turns are updating. (C) The UI refreshes with the updated conversation constraint.

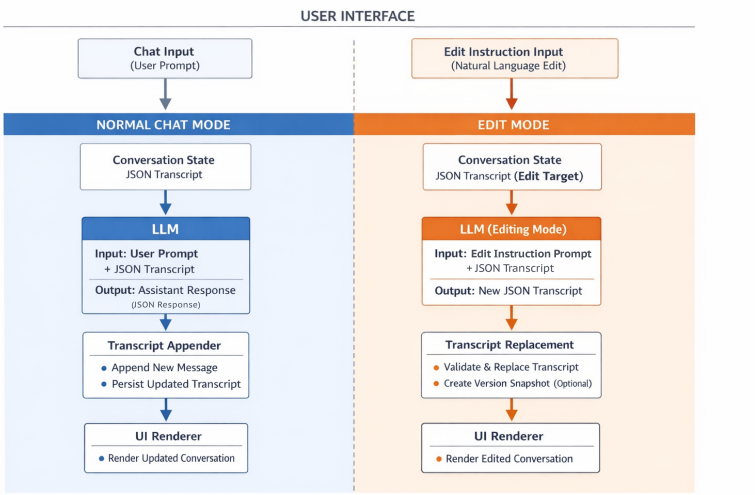


Figure 2: System overview of mutable transcripts operation. In normal chat mode, user prompts are appended to an immutable JSON transcript. In edit mode, natural language edit instructions are applied to the full transcript, which is rewritten and replaced before being re-rendered in the chat interface.

In default mode, user inputs are appended to the transcript and the full history is passed to the model to generate a response, reflecting how current systems operate. In "edit history" mode, the system instead constructs a prompt that includes the full transcript along with instructions describing how it should be modified. The full prompt template used for transcript rewriting is provided in Appendix A. The model generates an updated transcript reflecting the requested changes, which replaces the original transcript in the interface.

Figure 2 shows this process. The revised transcript becomes the new conversational state for subsequent interactions, ensuring that outdated or contradictory information is removed rather than retained.

This architecture is model-agnostic and can be implemented with any LLM capable of following structured natural language instructions. In our prototype, we use Gemini 2.5 [14] for both default and edit history modes. The prototype interface was built using Google AI Studio [15].

While the current prototype performs full transcript rewriting for simplicity and generality, this design choice is not fundamental to the mutable transcript paradigm. In particular, the use of full regeneration ensures consistency across all updated turns but introduces computational overhead, as the number of output tokens scales with conversation length. More efficient implementations could instead operate on structured diffs, localized updates, or retrieval-based recomposition of conversational state, reducing both latency and generation cost. Exploring such mechanisms is an important direction for future work, particularly for scaling to longer conversations.

4 Transcript-Level Operations

The following scenarios illustrate representative classes of transcript-level operations enabled by mutable transcripts. Each scenario demonstrates how revising conversational history, rather than appending corrective turns, allows the system to maintain a more accurate and current representation of user intent.

4.1 Retroactive Correction

Refer to the conversation depicted in Figure 1. In this scenario, the user realizes a mistake from several turns earlier and wants to correct it without derailing the conversation. In "Edit History" mode, they enter a prompt such as: "Change my question about GDP to reflect the year 2022, not 2021, and update all subsequent responses to reflect this change." The system updates the transcript so that downstream messages align with the corrected year, and the user can immediately see the revised flow in context. This example is depicted in Figure 1. This case demonstrates how mutable transcripts enable transcript-level propagation of corrections, allowing a single edit to consistently update all affected downstream turns.

4.2 Context Pruning

Here, the user wants to remove a brief diversion that could pollute the conversation context later on. In "Edit History" mode, they might enter: "Remove the discussion about the economic impact of climate change." The system rewrites the transcript to excise that detour while preserving the rest of the thread. This case illustrates that transcript-level revision enables selective removal of irrelevant or misleading content, preventing obsolete context from influencing subsequent model responses.

4.3 Constraint Injection

In this example, the user wants to retroactively apply new constraints across the entire conversation. In "Edit History" mode, they enter a prompt such as: "Update the whole discussion to assume: no cloud services, only on-device, and must be WCAG AA compliant." The system revises the transcript so that all prior messages reflect the new constraints. This case illustrates that transcript-level revision enables global application of new constraints, allowing them to be enforced consistently across all prior and future turns without the need to restart the conversation.

Beyond these core scenarios, Appendix E elaborates on various additional transcript-level operations which become possible when the transcript is treated as an editable state representation rather than a fixed sequence of turns.

5 User Study

To evaluate current correction behaviours and the effectiveness of mutable transcripts, we conducted a controlled user study in which participants completed a series of tasks under two conditions: a

standard chat interface (Condition A) and the mutable transcript interface described above (Condition B). Each participant experienced both conditions across three tasks, followed by a set of Likert-scale questions assessing interaction quality, clarity, confidence, and usability. These tasks correspond directly to the three primary classes of transcript-level operations introduced in Section 4: retroactive correction, constraint injection, and context pruning. We deliberately employed structured, standardized scenarios to isolate the three core transcript-level operations and enable controlled comparison between interaction conditions. This design was intended to establish initial interaction feasibility and usability, rather than assess performance in open-ended, naturalistic conversations.

In total, seventeen participants (n = 17) took part in the study. Participation was voluntary and no compensation was provided. This study was approved by the appropriate institutional ethics review board. All participants provided informed consent prior to participation. The majority of participants (15/17) were technology professionals working in research or software development. See Appendix C for further participant background information. To control for order effects, a counterbalanced within-subjects design was used: ten participants experienced Condition A followed by Condition B (A→B), while seven experienced the reverse order (B→A) for all three tasks. This design allows for direct comparison between conditions while mitigating learning and carryover effects.

5.1 Survey Structure

The data collection was carried out using a structured survey instrument. The full survey and task instructions are provided in the Appendix B and summarised here. The opening section gathers anonymous information about the participant and their current usage patterns of LLM-based chat interfaces. The second section guides the participant through three real-world scenarios in which they interact with the chat app to achieve a specified goal, but later realize that they need to correct or constrain earlier assumptions within the chat. Participants completed each task under both conditions, using standard chat (Condition A) and the mutable transcript interface (Condition B). The three tasks are designed to test the following corrective operations:

1. **Retroactive Correction** - Correcting information from several turns ago which has arisen due to user error or LLM misunderstanding.
2. **Constraint Injection** - Retroactively placing conditional constraints on a conversation to focus LLM responses more specifically on an outcome, e.g., budgetary or technical constraints.
3. **Context Pruning** - Retroactively removing outdated concepts or assumptions which are steering the conversation in the wrong direction.

After undertaking each task, participants are asked five follow-up questions about their experience under Condition A and Condition B separately. The questions are as follows:

1. **Transcript Cleanliness** *"This conversation thread is clean and free of outdated information that could mislead future chatbot responses."* Please indicate your level of agreement with this statement. Likert scale (1–7).
2. **State Clarity** *"It is easy to understand what assumptions or constraints are currently in effect within the conversation."* Please indicate your level of agreement with this statement. Likert scale (1–7).
3. **Confidence** *"I am confident the chatbot will follow the updated constraints going forward in a long conversation."* Please indicate your level of agreement with this statement. Likert scale (1–7).
4. **Restart Intent** *"If this were a real interaction with a chatbot, how likely would you be to start a new conversation instead of continuing with this conversation."* Please indicate your preference. Likert scale (1–7) (Very Unlikely – Very Likely). In this case, lower is better.
5. **Ease of Use** *How easy was it to apply your corrections?* Please indicate your preference. Likert scale (1–7) (Very Difficult – Very Easy).

For the restart-intent question, participants were instructed to assume they intended to continue working on the same topic, as detailed in Appendix B.

5.2 Results

The study included 17 participants. Across the combined dataset, Condition B consistently outperformed Condition A on all five evaluation metrics: transcript cleanliness, state clarity, user confidence, restart intent, and ease of use (Figure 3). Differences between conditions were statistically significant across all measures (all $p < 0.004$). Statistical significance was assessed using two-sided Wilcoxon signed-rank tests comparing Condition A and Condition B for each question, with responses averaged across the three tasks for each participant before performing paired comparisons. Error bars in Figure 3 represent 95% confidence intervals computed over participant-level mean responses.

Participants reported higher perceived transcript cleanliness and clarity under Condition B, indicating that the resulting transcript was easier to follow and better aligned with their intentions. This is particularly useful when a conversation is revisited later. Confidence scores were also significantly higher, suggesting that users felt more assured in the correctness and relevance of the system's outputs. Ease of use improved correspondingly, reflecting reduced interaction friction.

Notably, restart intent was lower in Condition B, indicating that participants were less likely to abandon the conversation and start over when transcript editing capabilities were available. These findings suggest that enabling direct revision of conversational history improves both perceived interaction quality and users' confidence in the system's adherence to updated intent.

The observed improvements in Condition B were consistent across both counterbalanced groups (A→B, n = 10; B→A, n = 7). In both orderings, participants showed more favourable scores for Condition B across all metrics, indicating that the direction of the observed differences was consistent across the two counterbalanced groups.

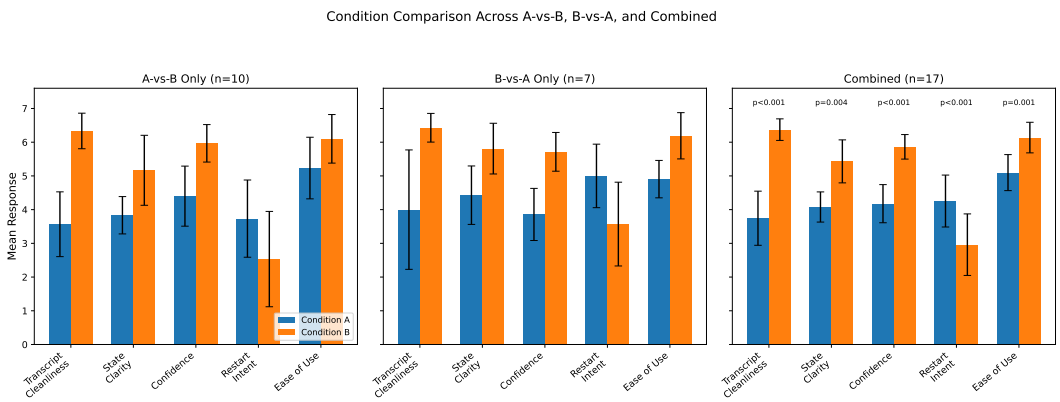


Figure 3: Condition A vs. B combined preference for five evaluation metrics: transcript cleanliness, state clarity, confidence, restart intent, and ease of use. The direction of the observed differences is consistent across both counterbalanced groups. The combined results show statistically significant preference for Condition B (mutable) vs. A across all questions, including a lower restart preference. Error bars represent 95% confidence intervals over participant-level mean responses

5.3 Transcript Analysis

To complement the user study, we performed a paired transcript analysis on representative instances of the three task scenarios (retroactive correction, constraint injection, and context pruning) evaluated in the user study. This enables comparison between subjective outcomes and the structural properties of the resulting transcripts. Each pair corresponds to the same instructed interaction carried out under standard chat conditions (immutable history) and with mutable transcripts enabled. These examples are intended to illustrate the structural impact of transcript mutability under controlled conditions, rather than to provide a large-scale benchmark. The conversation transcripts can be assessed in Appendix D.

Across all scenarios, Table 1 shows that mutable transcripts consistently reduced both conversation length and token load while eliminating obsolete retained context. On average, transcript length decreased from 8 to 4 turns (-50%), and total tokens decreased from 581 to 238 (-59%). In contrast,

Table 1: Transcript efficiency comparison between standard immutable chat histories (Baseline) and mutable transcripts (Mutable) across three representative revision scenarios. Mutable transcripts reduce conversation length and eliminate obsolete retained context arising from superseded corrections and constraints.

Scenario	Method	Turns	Tokens	Obsolete Tokens	Obsolete %
Retroactive Correction	Baseline	8	422	236	56%
	Mutable	4	**169**	–	–
Constraint Injection	Baseline	6	608	353	58%
	Mutable	4	**479**	–	–
Context Pruning	Baseline	10	712	655	92%
	Mutable	4	**66**	–	–
Mean	Baseline	8	581	415	71%
	Mutable	4	**238**	–	–
Turn/Token Reduction		**50%**	**59%**	–	–

baseline chat histories retained a substantial proportion of obsolete context, with an average of 71.4% of tokens corresponding to superseded or no-longer-relevant information. In Table 1, "Obsolete Tokens" is calculated as the token total of all turns (user and assistant) which have been superseded by a subsequent corrected turn. Token counts were computed using the `tiktoken` tokenizer [16].

These results provide a quantitative illustration of how transcript structure differs between standard chat histories and mutable transcripts in representative scenarios. In particular, they highlight the extent to which obsolete context can persist in baseline interactions and the relative compactness of transcripts after revision. While limited in scope, this analysis is intended to give a concrete sense of these structural differences and the magnitude of retained obsolete context, rather than to establish generalizable performance improvements.

5.4 Correction Method Preference

Prior to undertaking the tasks, participants were asked how they typically correct mistakes or refine inputs in existing chat interfaces. As shown in Figure 4, the most common strategy is sending a follow-up message containing the correction (94.1%), followed by copying and modifying a previous prompt before resending it (70.6%). Direct editing of previous messages, where available, is less commonly used (35.3%), and restarting the conversation entirely is relatively uncommon (23.5%). Only a small minority of participants reported not correcting previous inputs at all (5.9%).

After completing the tasks, 88.2% of participants indicated that they would use an "Edit History" feature if it were available in real chat interfaces. This suggests a strong unmet need for more direct and flexible mechanisms for revising conversational context beyond editing the most recent message.

6 Discussion

The results highlight several key insights. First, users currently rely on indirect and often inefficient strategies, such as follow-up corrections and prompt rewriting, to manage evolving intent in chat interfaces. These behaviors reflect the limitations imposed by immutable conversation histories.

Second, providing users with the ability to revise earlier parts of the transcript is associated with measurable improvements in interaction quality. Participants experienced clearer, more coherent conversations, reported greater confidence in system outputs, and found the interface easy to use.

Third, the reduction in restart intent suggests that mutable transcripts may help preserve conversational continuity. Rather than abandoning a thread due to accumulated errors or misalignment, users can repair and refine the existing conversation.

Taken together, these findings support the view that immutable chat histories can lead to the retention of outdated or misaligned context, and that structured, user-driven revision of conversation history can improve usability while maintaining a cleaner conversational state. Mutable transcripts are not primarily intended to improve immediate task success in short interactions. Rather, they are intended

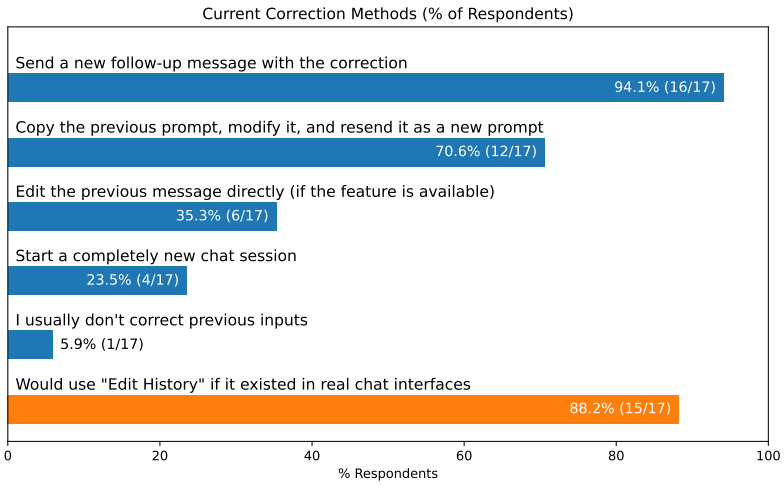


Figure 4: Current correction methods used by respondents (blue) and the percentage of respondents who would use a mutable transcript feature if it was available (orange). The majority of people (94.1%) send a follow-up message to perform corrections and 88.2% of people said they would use "Edit History" if it existed in a real chat interface.

to preserve conversational progress while reducing accumulated context burden, obsolete retained state, and transcript growth over iterative correction cycles.

7 Conclusions and Future Work

We introduced mutable transcripts as an interaction paradigm for AI chat interfaces in which conversation history is treated as an editable representation of conversational state. By allowing users to revise prior turns through natural language edit requests, this approach enables more flexible management of evolving intent while preserving the conversational interaction model. We demonstrated the feasibility of this paradigm through a working prototype, structured case studies, and a controlled user study.

Our findings indicate that participants rated mutable transcripts more positively than standard chat across measures of clarity, confidence, and ease of use, and reported lower intent to restart conversations when revision capabilities were available. Complementary transcript analysis provided a quantitative illustration of how conversational structure differs under each interaction model, highlighting the extent to which obsolete context can persist in baseline interactions. However, the small, predominantly technically experienced participant sample limits the generalizability of the observed usability preferences to broader user populations.

Mutable transcripts also introduce important challenges. Allowing prior messages to change complicates authorship, accountability, and trust, making it less clear why a system's behavior has shifted or which assumptions underlie particular responses. Natural language edit requests are inherently ambiguous, and models may misinterpret user intent or apply changes inconsistently. In our study, some participants observed that useful information could be lost during revision. Revising earlier turns may also require users to re-examine affected subsequent responses, introducing a potential cognitive burden that was not measured in our study. Addressing these issues may require interface and system-level safeguards, such as previews, undo functionality, version histories, or visual diff mechanisms. Another key practical consideration is the trade-off between transcript consistency and generation cost, as full-transcript rewriting simplifies global updates but may introduce latency for longer conversations. Finally, mutable transcripts may be inappropriate in contexts requiring immutable records, such as legal or audit-sensitive domains.

Overall, these results suggest that enabling structured, user-driven revision of conversational history is a promising direction for improving human-AI interaction. Rather than replacing existing approaches to context management, mutable transcripts offer a complementary model for maintaining a more current and user-aligned representation of conversational state. While the present evaluation demonstrates improvements in perceived interaction quality and illustrates reductions in obsolete

retained context, further work is needed to establish whether these benefits translate into measurable improvements in conversational state management and downstream task performance. Future work should also explore more robust mechanisms for controlled editing, evaluate the approach in longer and more naturalistic conversations, and investigate integration with system-level context management techniques.

Acknowledgments

This publication has emanated from research conducted with the financial support of Taighde Éireann – Research Ireland under Grant No. 23/RC/13506 at the Research Ireland Centre – Rinn Artificial Intelligence, and with support from a gift from Google. For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission. The authors declare no competing interests.

References

[1] Shiri Melumad and Jin Ho Yun. Experimental evidence of the effects of large language models versus web search on depth of learning. *PNAS Nexus*, 4(10):pgaf316, 2025. ISSN 2752-6542. doi: 10.1093/pnasnexus/pgaf316. URL https://doi.org/10.1093/pnasnexus/pgaf316.

[2] Ruimiao Li, Manli Li, and Weifeng Qiao. Engineering students' use of large language model tools: An empirical study based on a survey of students from 12 universities. *Education Sciences*, 15(3):280, 2025. doi: 10.3390/educsci15030280. URL https://www.mdpi.com/2227-7102/15/3/280.

[3] Weixin Liang, Yaohui Zhang, Mihai Codreanu, Jiayu Wang, Hancheng Cao, and James Zou. The widespread adoption of large language model-assisted writing across society. *Patterns*, 6 (12):101366, 2025. doi: 10.1016/j.patter.2025.101366. URL https://doi.org/10.1016/j.patter.2025.101366.

[4] Katie Seaborn. Chatgpt and u(x): A rapid review on measuring the user experience, 2025. URL https://arxiv.org/abs/2503.15808.

[5] Davide Baldelli, Ali Parviz, Amal Zouaq, and Sarath Chandar. Llms can't play hangman: On the necessity of a private working memory for language agents, 2026. URL https://arxiv.org/abs/2601.06973.

[6] OpenAI. Conversation state - managing multi-turn context. https://platform.openai.com/docs/guides/conversation-state.

[7] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts, 2023. URL https://arxiv.org/abs/2307.03172.

[8] Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. LLMs get lost in multi-turn conversation, 2025. URL https://arxiv.org/abs/2505.06120.

[9] Jeff Wu, Long Ouyang, Daniel M. Ziegler, Nisan Stiennon, Ryan Lowe, Jan Leike, and Paul Christiano. Recursively summarizing books with human feedback, 2021. URL https://arxiv.org/abs/2109.10862.

[10] Qingyue Wang, Yanhe Fu, Yanan Cao, Shuai Wang, Zhiliang Tian, and Liang Ding. Recursively summarizing enables long-term dialogue memory in large language models, 2025. URL https://arxiv.org/abs/2308.15022.

[11] 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. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2020. URL https://arxiv.org/abs/2005.11401.

[12] OpenAI. Introducing Canvas. https://openai.com/index/introducing-canvas/, 2024.

[13] Fábio Perez and Ian Ribeiro. Ignore previous prompt: Attack techniques for language models, 2022. URL https://arxiv.org/abs/2211.09527.

[14] Google. Google AI for Developers - Gemini API Documentation. https://ai.google.dev/gemini-api/docs.

[15] Google. Google AI Studio. https://aistudio.google.com/apps, 2024.

[16] OpenAI. tiktoken: Fast BPE tokeniser for OpenAI models. https://github.com/openai/tiktoken, 2023.

Appendix

A Full System Prompt

A.1 Prompt Template for Edit History Mode

You are a conversation editor. Here is the current conversation history in JSON format:

`$JSON.stringify(history.map(m => ( role:  m.role, text:  m.text )))`

User Instruction: `"$instruction"`

Task: Rewrite the conversation history above to satisfy the user instruction. You can modify, delete, or add messages as needed to make the history consistent with the instruction.

Critical: You must maintain all Markdown formatting (like lists, bold, italics, code blocks, and headers). If the original message used a list, the rewritten message must also use a list unless the instruction explicitly says otherwise. Ensure that list items start with a newline and a proper bullet point (e.g., "* Item").

For example, if the instruction is "Make us sound like pirates", rewrite all messages in pirate speak while keeping any list structures. If the instruction is "Remove the second message", remove it and ensure the flow still makes sense.

Return the new history as a JSON array.

A.2 Prompt Template for Default Chat Mode

You are a helpful and concise AI assistant.

B User Study Design and Materials

This appendix provides the full survey instrument used in our user study, including task instructions, interface conditions, and post-task evaluation questions. The survey was designed to evaluate user interaction with the system across three representative scenarios—retroactive correction, constraint injection, and context pruning—under both a standard chat interface and the proposed method. Participants completed each task in both conditions, followed by Likert-scale questions assessing transcript cleanliness, state clarity, confidence, restart intent, and ease of use. The participants engaged in the study voluntarily and were not compensated for the task. The materials included here are intended to support reproducibility and provide transparency into the experimental design and evaluation protocol.

Google Forms was used as the survey instrument. The full survey and participant information sheets can be accessed in the supplementary materials zip file attached to this submission. For convenience, the contents of the supplementary zip file can also be browsed online:

`https://mutable-transcripts.netlify.app/`

B.1 Example Task Instructions Given to Participants

We provide here one example of the task instructions given to the participants. Instructions for all three tasks can be accessed in the full user study survey contained in the supplementary zip file or link above.

Task 1A - Retroactive Correction (without Edit History)

Scenario. You are carrying out research into the GDP of a certain country. You ask an LLM and engage in a conversation.

Please follow these steps accurately. Do **NOT** use “Edit History” mode in this task yet.

1. Go to the chatbot tab in your browser
   (Source code and prototype here: `https://github.com/QxLabIreland/ReChat`)

2. Start a new chat using the "**New Chat**" button
3. Ask: **What was the GDP of Ireland in 2021?**
4. Read the response
5. Ask: **What are the largest categories?**
6. Read the response
7. You realise you made a mistake. Ask: **Sorry, I meant Iceland, not Ireland**
8. Read the response
9. You realise you made another mistake. Ask: **Sorry, I meant 2022**
10. Read the response
11. Review the thread briefly. Just scan through it.

Imagine you intend to continue this conversation for further analysis of Iceland's GDP in 2022. You will now be asked some questions about this conversation experience.

Task 1B - Retroactive Correction (with Edit History)

You will now carry out the same task using "**Edit History**" mode as instructed.

Scenario: you are carrying out research into the GDP of a certain country. You ask an LLM and engage in a conversation.

Please follow these steps accurately. This time you will use "**Edit History**" mode as instructed.

1. Go to the chatbot tab in your browser
   (Source code and prototype here: `https://github.com/QxLabIreland/ReChat`)
2. Start a new chat using the "**New Chat**" button
3. Ask: **What was the GDP of Ireland in 2021?**
4. Read the response
5. Ask: **What are the largest categories?**
6. Read the response
7. You realise you made a mistake. Enable "**Edit History**" mode and Ask: **Sorry, I meant Iceland, not Ireland**
8. Read the updated responses
9. You realise you made another mistake. Enable "**Edit History**" mode and Ask: **Sorry, I meant 2022**
10. Read the updated response
11. Review the thread briefly. Just scan through it.

Imagine you intend to continue this conversation for further analysis of Iceland's GDP in 2022. You will now be asked some questions about this conversation experience.

Likert Questions Asked After Each Task

After undertaking each task, participants are asked five follow-up questions about their experience under Condition A and Condition B separately. The questions are as follows:

1. **Transcript Cleanliness** *"This conversation thread is clean and free of outdated information that could mislead future chatbot responses."* Please indicate your level of agreement with this statement. Likert scale (1–7).
2. **State Clarity** *"It is easy to understand what assumptions or constraints are currently in effect within the conversation."* Please indicate your level of agreement with this statement. Likert scale (1–7).

3. **Confidence** *"I am confident the chatbot will follow the updated constraints going forward in a long conversation."* Please indicate your level of agreement with this statement. Likert scale (1–7).
4. **Restart Intent** *"If this were a real interaction with a chatbot, how likely would you be to start a new conversation instead of continuing with this conversation."* Please indicate your preference. Likert scale (1–7) (Very Unlikely – Very Likely). In this case, lower is better.
5. **Ease of Use** *How easy was it to apply your corrections?* Please indicate your preference. Likert scale (1–7) (Very Difficult – Very Easy).

C User Study Data and Results

C.1 Per-Participant Results

Here we provide access to the raw, per-participant data collected in the user study as a CSV file. The dataset includes individual responses across all tasks and evaluation questions, enabling detailed inspection of participant-level variation and supporting reproducibility and transparency of the reported results. The results are available in the supplementary zip file submitted with this paper. For convenience, the contents of the supplementary zip file can also be browsed online:

`https://mutable-transcripts.netlify.app/`

C.2 Qualitative Feedback from Participants

Participants were given the option to comment on their experiences after each task. The following sections summarise both the positive and negative feedback received.

Participant comments on Condition B (Edit History) were broadly positive. Respondents frequently described the rewritten conversations as cleaner, less cluttered, easier to reread, and more manageable overall. Several comments suggested that Rewrite History was particularly useful when the task was well defined or when the conversation had become unnecessarily long, with some participants noting that it made the thread feel substantially shorter and easier to review. A small number of comments also described the feature as quick and easy to use.

Negative feedback on Condition B focused less on outright failure and more on trade-offs introduced by transcript rewriting. Several participants expressed concern that overwriting earlier turns removed potentially useful original information, making it harder to recover prior versions of the conversation or compare before-and-after states. Related comments suggested a need for better transparency and control, such as diff views, persistent edit mode behavior, clearer reminders of active constraints, or some form of version history. A few participants also felt that Rewrite History was not always necessary for more exploratory conversations, where preserving the original thread could still be valuable.

C.3 Additional Survey Results

This section provides supplementary descriptive statistics from the user study survey. The tables report participant demographics, usage patterns, and self-reported behaviors related to chatbot interaction and correction strategies. These results complement the main paper by offering additional context on the participant population and their prior experience with LLM-based systems.

Table 2: Participant professional or academic field ($N = 17$).

Profession Type	Count	Percent
Researcher	9	52.9
Data Science / Machine Learning	4	23.5
Software Development	2	11.8
Other (Specified "Lecturer")	1	5.9
Business/Management	1	5.9

Table 3: Frequency of LLM chatbot use for work, study, or personal tasks ($N = 17$).

Response	Count	Percent
Multiple times a day	9	52.9
Several times a week	5	29.4
Once a day	2	11.8
Less than once a week	1	5.9

Table 4: LLM chatbots respondents were most familiar with or used most often ($N = 17$ respondents; multi-select).

Option	Count	Percent
Gemini	16	94.1
ChatGPT (OpenAI)	12	70.6
Claude (Anthropic)	3	17.6
Microsoft Copilot	3	17.6
Other	3	17.6

Percentages are calculated over respondents rather than selections because participants could choose more than one option.

Table 5: How often respondents felt the need to edit or modify a previous chatbot message or prompt ($N = 17$).

Response	Count	Percent
Sometimes	8	47.1
Frequently	6	35.3
Never	2	11.8
Always or almost always	1	5.9

Table 6: Current actions taken when changing a previous chatbot input ($N = 17$ respondents; multi-select).

Option	Count	Percent
Edit the previous message directly (if the feature is available)	6	35.3
Send a new follow-up message with the correction	16	94.1
Copy the previous prompt, modify it, and resend it as a new prompt	12	70.6
Start a completely new chat session	4	23.5
I usually don't correct previous inputs	1	5.9

Percentages are calculated over respondents rather than selections because participants could choose more than one option.

Table 7: Whether respondents had encountered similar situations in real chatbot use ($N = 17$).

Response	Count	Percent
Sometimes	6	35.3
Very often	5	29.4
Often	4	23.5
Rarely	2	11.8

Table 8: Likelihood of using an "Edit History" feature if available in real chatbots ($N = 17$).

Response	Count	Percent
Likely	9	52.9
Very Likely	6	35.3
Neutral	1	5.9
Unlikely	1	5.9

D Transcript Analysis Data

To complement the illustrative analysis presented in Section 5.3, we provide access to the full conversation transcripts used in the transcript-level comparisons. These consist of one representative pair of conversations per scenario (retroactive correction, constraint injection, and context pruning), with each pair showing the same task carried out under both conditions: standard append-only chat (Condition A) and mutable transcripts (Condition B).

These transcripts are provided as CSV files in the supplementary zip file submitted with the paper. For convenience, the contents of the supplementary zip file can also be browsed online:

`https://mutable-transcripts.netlify.app/`

Each transcript includes:

1. The full sequence of user and assistant messages
2. The corresponding revised transcript (for the mutable transcript condition)
3. Token counts computed using the tiktoken tokenizer
4. Annotation of obsolete tokens, defined as tokens belonging to turns superseded by later corrections or constraints

We emphasize that this analysis is illustrative rather than exhaustive. The selected transcript pairs are intended to demonstrate representative structural effects of transcript revision under controlled conditions, rather than to reflect aggregate behavior across all study participants. As such, the results should be interpreted as qualitative examples that complement the user study findings, not as a statistically powered evaluation.

Future work may extend this analysis to larger datasets and automated benchmarks to more systematically quantify the impact of transcript revision on context efficiency and downstream model performance.

E Additional Transcript Operations

These examples highlight additional operations that become possible when the transcript is treated as an editable state representation rather than a fixed sequence of turns.

1. **Conversation Simulation and Expansion:** In "Edit History" mode, users can simulate additional turns to expand the conversation beyond its current state. This enables creative exploration of topics, as the LLM generates both questions and answers for the new turns. Potential applications include argument or debate simulation, interview preparation, and other scenarios where extending dialogue is valuable.
2. **Emotional Reframing:** Users can retrospectively experiment with simulating different emotional tones in the conversation. By using "Edit History" mode to prompt "Change the emotional tone of the conversation to be more (optimistic, angry, surreal, etc)," the LLM can reframe previous messages (user or AI) to reflect a different emotional perspective, allowing for exploration of how emotional context influences dialogue.
3. **Language Translation:** Users can experiment with retrospectively changing the language of the entire conversation to a different foreign language. By using "Edit History" mode to prompt "Translate the conversation into Spanish (or any other language)," the LLM can rephrase all previous messages in the selected language, enabling multilingual exploration and communication.
4. **Style Transfer:** Users can also experiment with changing the style of the language used in the conversation retrospectively. By using "Edit History" mode to prompt "Change the language style to be more formal/informal," the LLM can adjust the tone and style of all previous messages to match the new style, allowing for exploration of different communication styles.
5. **Redaction and Controlled Sharing:** If a user wants to share a conversation that contains sensitive details, names, emails, internal URLs, or project codenames, they can use "Edit

History" mode to request redaction. A prompt like "Redact all personal and confidential details, replace them with consistent placeholders (e.g., [NAME_1], [ORG_1]), and update the rest of the conversation so it still reads coherently" produces a safe-to-share transcript. This case illustrates that transcript-level revision enables structured redaction of sensitive information while preserving overall coherence, supporting safe sharing of conversational artifacts.

6. **Transcript Import:** Users can load external conversations into the current chat context. By using "Edit History" mode to prompt "Load the following conversation into the context, [conversation text, JSON, MD, etc]" it is possible to load previous conversations exported from other platforms and use the Edit History features on them too.

F Limitations

F.1 Computational Overhead of Transcript Rewriting

The current implementation requires regenerating the full transcript during edit operations, which introduces additional latency and computational cost compared to standard next-turn generation. Because output tokens scale with conversation length, this approach may become inefficient for longer interactions. While revised transcripts are often more compact for subsequent turns, this benefit comes with a trade-off during editing. Future implementations could reduce this overhead by operating on structured diffs, partial updates, or retrieval-based reconstruction of conversational state.

F.2 Branching Comparison

We do not include a direct empirical comparison with branching-based interfaces, as branching represents a complementary interaction model that preserves prior context rather than revising it. Evaluating trade-offs between branching and revision-based interaction paradigms like mutable transcripts is an important direction for future work.

F.3 Loss of Useful Context

Although rewriting can reduce clutter and remove obsolete information, it can also discard context that remains useful. Earlier turns may contain alternative ideas, abandoned directions, or intermediate reasoning that users later want to revisit. When these are overwritten, the interaction becomes cleaner but less recoverable, making it harder to compare versions of the conversation or restore a previously useful state. This suggests that transcript rewriting may benefit from complementary mechanisms such as version history, undo, or diff-based views.

F.4 Security Risks

Editable transcripts introduce additional security and trust challenges beyond standard append-only chat systems. In particular, malicious or misleading instructions could be inserted retroactively into earlier conversational turns, potentially influencing future model behaviour in unintended ways, including attempts to override prior constraints or manipulate alignment behaviour. Mutable history may also make it harder for users to understand which assumptions or edits led to a given response.

The current prototype does not attempt to address these risks directly. Instead, this work focuses on the interaction and usability implications of editable conversational state. Future systems may require safeguards such as version histories, diff views, provenance tracking, or constrained editing policies to support trustworthy deployment.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the paper's main contributions—introducing the concept of mutable transcripts, presenting a prototype system, and evaluating it through a user study. These claims are directly supported by the content of the paper, including the system design, empirical evaluation, and analysis of interaction outcomes.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper explicitly discusses several limitations of the proposed approach. These include challenges around ambiguity in natural language edit requests, potential loss of useful information during transcript revision, and issues related to trust, accountability, and transparency when conversation history can be modified. It also notes that mutable transcripts may be inappropriate in certain domains (e.g., legal or audit-sensitive settings) and highlights the limited scale of the user study.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not present formal theoretical results such as theorems or proofs. Its contributions are primarily conceptual and empirical (prototype design and user study evaluation), so the requirement for formal assumptions and proofs does not apply.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides sufficient detail to reproduce the main experimental results given the nature of the contribution (a user study and prototype system). It clearly describes the study design (tasks, conditions, counterbalancing, and evaluation metrics), as well as the system setup using a hosted LLM API. Full source code and README are available at: `https://github.com/QxLabIreland/ReChat`.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The paper provides access to a live, hosted version of the system for anonymous review purposes. This allows reviewers to directly interact with and evaluate the method and reproduce the qualitative and user study conditions described in the paper. Full source code will be made publicly available upon acceptance, including full implementation details and API configuration. This provides a reasonable and practical pathway for reproducibility during review, with full open access planned post-acceptance.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper provides sufficient detail about the experimental setup for its type of study (a controlled user study and prototype evaluation), including task design, conditions (standard vs. mutable transcripts), participant structure, and evaluation metrics (Likert-scale questions and measured outcomes). Since the work does not involve model training, there are no hyperparameters, optimizers, or data splits to report.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.

- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The paper reports statistically significant differences between conditions and includes appropriate statistical details, including 95% confidence intervals and two-sided Wilcoxon signed-rank tests using participant-level mean responses averaged across the three tasks. The factors of variability (participant-level responses in a within-subjects design) are clear, and the statistical testing methodology is specified.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The experiments are conducted using a hosted API (Gemini 2.5) rather than local or user-managed compute infrastructure. As a result, compute resources such as hardware type, memory, and execution time are abstracted by the service provider and are not directly controlled by the authors. A publicly accessible hosted version of the system and full source code are available at: `https://github.com/QxLabIreland/ReChat`.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The research adheres to the NeurIPS Code of Ethics. The study involves a small-scale user study with no indication of harmful, deceptive, or unethical practices, and it respects participant interaction norms. The paper contains no identifying participant information and does not involve sensitive data collection, high-risk deployments, or ethically problematic methodologies.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The paper discusses both positive and negative societal impacts. On the positive side, it highlights how mutable transcripts can improve human–AI interaction by reducing context pollution, increasing clarity, and helping users maintain alignment with their evolving intent, which can enhance productivity and accessibility of AI systems. On the negative side, the paper explicitly acknowledges risks such as reduced transparency and accountability when past interactions can be modified, potential misuse through prompt injection or manipulation of conversational history, and challenges around trust and auditability of AI outputs. It also notes that such systems may be inappropriate in high-stakes or regulated settings (e.g., legal or audit contexts).

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not introduce or release any high-risk or dual-use assets such as large pre-trained models, generative systems, or scraped datasets. The contributions are limited to a prototype interface and a small-scale user study, neither of which poses significant misuse risks.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [N/A]

Justification: The paper does not rely on any external datasets, codebases, or pre-trained models as core research assets. Instead, it uses a self-collected user study dataset and a prototype system developed by the authors. Therefore, there are no third-party assets that require licensing, attribution, or terms-of-use compliance.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: All of the source code, documentation and a demo website can be accessed at `https://github.com/QxLabIreland/ReChat`

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.

- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [Yes]

    Justification: This research contains a user study which involved human participants (n=17). The full text of the survey including user study instructions is included in the supplementary material.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [Yes]

    Justification: This study involved human participants (n=17). Participants provided informed consent. The study was conducted in accordance with institutional guidelines. Human Research Ethics Committee approval was obtained as required by institutional policy. Participants were not compensated.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: The prototype system uses Gemini 2.5 to implement conversational transcript revision through natural-language instructions, as described in Section 3.1. The prompts used for transcript rewriting are provided in Appendix A.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.