that better models require less prompt engineering, as they are more robust to minor variations in inputs and outputs [47].

The sharded setting paints a different picture. Model aptitude degrades in a non-significant way between the full and sharded settings, with an average drop of 16%. On the other hand, unreliability skyrockets with an average increase of 112% (more than doubling). More interestingly, though better models tend to have slightly higher multi-turn aptitude, all models tend to have similar levels of unreliability. In other words, **in multi-turn, underspecified settings, all models we test exhibit very high unreliability, with performance degrading 50 percent points on average between the best and worst simulated run for a fixed instruction.** This refines our definition of the *lost in conversation* phenomenon: when comparing single- and multi-turn settings, we find that large performance degradations ($\overline{P}$) are due in large part to increased model unreliability (U), rather than a loss in aptitude (A).

Appendix F explores potential root causes for models getting lost in conversations. We identify four specific causes: (1) LLMs prematurely propose full answer attempts, making assumptions about problem specifications that lead to confusion (Appendix F.1), (2) they overly rely on previous (incorrect) answer attempts leading to lengthier "bloated" answers (Section F.2), (3) LLMs overly adjust their answers based on the first and last turn of conversation, evidenced by a loss-of-middle-turns phenomenon (Appendix F.3), and (4) they produce overly verbose answers, which likely introduces assumptions that detract attention from user utterances (Section F.4).

6.3 Gradual Sharding Experiment

The multi-turn conversations simulated based on sharded conversations are not representative of underspecified conversations that users might have with LLMs in realistic settings. In particular, the fact that sharded instructions must be maximal (property P3) and that the simulated user must reveal at most one shard of information per turn (Section 3.2) can seem unrealistic and adversarial. In fact, prior work has found that minor and severe underspecification appear in equal proportions in public LLM chat logs [27]. To explore the relationship between the granularity of sharding and the lost in conversation phenomenon, we propose the gradual sharding experiment.

In the gradual sharding experiment, we selected 31 instructions from our original experiment across multiple tasks, and expanded each sharded instruction into seven sharded instructions, with the shard-set size growing from 2 to 8 shards. The instruction selection and sharding process are detailed in Appendix K. The process ensured that at each shard set size (from 1 to 8), task complexity is fixed, and the only modified factor is the granularity of sharding.

We ran simulations for the gradual sharding experiments with two models (GPT-4o and GPT-4o-mini), with results summarized in Figure 6c. We find that both models get lost in conversation (a minor degradation in aptitude and a large increase in unreliability) with two-shard instructions and beyond. In other words, the gradual sharding experiment indicates that **any conversation that involves underspecification and occurs in two or more turns leads to models getting lost in conversation**. For users, the granularity at which information is specified does not majorly impact reliability: providing all the information at once (1-shard) is the only effective method to improve reliability.

7 Implications

7.1 Implications for System and Agent Builders

	Simulation Type				
Model	Full	Concat	Sharded	Recap	Snowball
4o-mini	86.8	84.4	50.4	66.5	61.8
4o	93.0	90.9	59.1	76.6	65.3

Table 2: Experimental Results with additional simulation types: Recap and Snowball. Both strategies involve repeating user-turn information to mitigate models getting lost in conversations.

Building LLM-based applications typically involves complex processes: decomposition of problems, retrieval of relevant information, use of tools, and calling of actions. Such processes are typically orchestrated by an agentic framework (such as Autogen [84] or LangChain [8]) that allows system builders to compose workflows with LLM calls as individual blocks. As such, an argument could be made that multi-turn capabilities are not a necessary feature of LLMs, as it can be offloaded to the agent framework. In other words, do we need native multi-turn support in LLMs when an agent framework can orchestrate interactions with users and leverage LLMs only as single-turn operators?

To answer this question, we implemented two agent-style conversation simulation types: RECAP and SNOWBALL. Both preprocess user utterances before sending them to the LLM. In RECAP, a conversation proceeds in the same way as SHARDED, but a user turn is added at the end, which recapitulates all the previous user turns. SNOWBALL is a more gradual recapitulation: at each turn, the user simulator reveals a new shard, and repeats all previously revealed shards at that point. Both simulation types repeat the past user's turn information to make it more prominent and give the LLM a chance to leverage the redundancy to improve its responses. We include the experimental detail in Appendix M.

Table 2 summarizes the results on all instructions for four tasks (Code, Database, Math, Actions) for two tested models (GPT-4o, GPT-4o-mini). Both RECAP and SNOWBALL demonstrate some level of success, with improvements over SHARDED simulations, but the performance still lags behind FULL or CONCAT. While RECAP outperforms SNOWBALL, we note that RECAP is an unrealistic setting because the intervention is conducted on the *last turn* of the conversation, which is not known a priori when conversation unfolds with a real user. SNOWBALL gives a sense of realistic performance gains achievable through user-turn repetition: it can mitigate the FULL-to-SHARDED performance deterioration by 15-20%. In short, relying on an agent-like framework to process information might be limiting, and we argue LLMs should natively support multi-turn interaction.

7.2 Implications for LLM Builders

A lot of effort has been put in improving LLM *aptitude*: demonstrating that LLMs can accomplish tasks of increasing intellectual complexity, with recent results showing LLMs can compete in mathematics Olympiads, or solve Ph.D.-level technical questions in a benchmark aptly named Humanity's Last Exam [62].

In this work, we call on LLM builders to prioritize *reliability* of the models they build, as our experiments demonstrate that the randomness involved in generating text with LLMs leads to catastrophic unreliability in all the models we tested, degrading the quality of responses the average LLM users see.

LLMs are probabilistic systems, with parameters such as *temperature* that can adjust the degree of randomness that occurs while generating text. A possible argument is therefore: does setting the temperature to its lowest setting ($T = 0$) effectively resolve the reliability concern, as it makes the generation process more (but not entirely) deterministic?

To evaluate this argument, we conducted a supplementary experiment in which the assistant's temperature for generating responses (AT) was varied to three values: 1.0, 0.5, and 0.0. Additionally, since SHARDED simulation uses an LLM-based user simulator, we also varied the user's temperature (UT) with the same three values. Further details on the experiment, including sample selection and simulation scale, are in Appendix L.

Table 3 summarizes the experimental findings. Looking at the FULL and CONCAT settings (first two rows), both GPT-4o-mini and GPT-4o observe a large improvement in reliability when temperature is decreased, with a drop in unreliability (U_{10}^{90}) of 50-80% when the assistant temperature decreases. Results from SHARDED simulations are more alarming: GPT-4o-mini does not see improvements in reliability as AT is decreased (in all user-temperature settings), and GPT-4o only sees minor improvements, on the order of 15-20%. Even when both the user and assistant temperatures are set to 0.0, there remains a large unreliability of around 30%. Even though language models are supposed to be deterministic at $T = 0.0$, this is known to practically not be the case for modern LLMs (see Appendix N for discussion). At a high level, single-turn conversations have limited scope for deviation, whereas one token difference in an early turn of a multi-turn conversation can lead to cascading deviations, which we observe as stagnated unreliability. For settings that involve multi-turn interaction, we find that **lowering the temperature of the LLM when generating responses is ineffective in improving system reliability.**

	4o-mini			4o		
Simulation	AT=1.0	AT=0.5	AT=0.0	AT=1.0	AT=0.5	AT=0.0
FULL	16.0	15.0	6.8	17.8	8.0	2.8
CONCAT	20.2	17.8	9.5	20.2	17.8	5.8
UT=1.0	49.8	46.8	51.0	41.0	43.8	31.8
UT=0.5	31.7	34.0	40.5	39.5	40.8	31.8
UT=0.0	38.5	28.0	30.5	35.8	38.0	29.7

Table 3: Unreliability of models when changing assistant temperature (AT) and user temperature (UT) in FULL, CONCAT and SHARDED settings. The lower the number the more reliable the assistant is.

We invite and challenge LLM builders to jointly optimize model aptitude and reliability. A reliable LLM should: (1) achieve similar aptitude in single- and multi-turn settings, (2) have small unreliability ($U_{10}^{90} < 15$) in multi-turn settings, (3) achieve these at unmodified temperature ($T = 1.0$), demonstrating that the underlying language model can handle variations that naturally occur in language generation.

7.3 Implications for NLP Practitioners

Our experiments demonstrate that model behavior in single- and multi-turn settings on the same underlying set of instructions can diverge in important ways, for example, with large observed degradations in performance and reliability.

We selected the initial six tasks to span a wide range of generation tasks, from programming to multi-document summarization. Yet this set of tasks is limited across multiple dimensions, such as focusing on English-language instructions and analytical (i.e., non-creative) tasks. We put effort into making the sharding process scalable by automating portions that could be handled by an LLM, while manually validating and finalizing samples for quality control. The sharding process – detailed in Appendix C – required an average of three hours of manual work (prompt engineering or inspection) from an author to prepare and finalize 100 sharded instructions.

We encourage NLP practitioners to experiment with sharding and release sharded versions of their tasks and instructions alongside fully specified ones.

	Translation		
Model			
4o-mini	41.7	43.4	42.1
4o	35.9	38.5	40.9

Table 4: Performance on the translation task for FULL, CONCAT, and SHARDED simulations.

To illustrate the feasibility of sharding new tasks, and understand compatibility requirements for sharding, we prepared sharded instructions for a seventh task: Translation. The task consists of translating an entire document (10 sentences) from German to English, leveraging paired documents from WMT 2019 on document-level translation [70]. In the SHARDED setting, each turn reveals two additional sentences from the source document and requires the assistant to translate all sentences provided so far, whereas the FULL and CONCAT settings reveal the entire document in the first turn. Evaluation is conducted with the standard BLEU metric [58]. We describe practical implementation details in Appendix I.

Results from FULL, CONCAT, and SHARDED simulations are summarized in Table 4. Both models we tested – GPT-4o-mini and GPT-4o – do *not* exhibit degradation in performance in the SHARDED setting, with BLEU scores being within 10% difference of each other in all settings. We believe this result reflects that the task can largely be accomplished at the sentence-level despite some prior work has framed translation at the document-level [64], and that the BLEU score does not adequately capture document-level nuances [52]. In other words, if a task is episodic (i.e., it can be decomposed into turn-level subtasks), the models can avoid getting lost in conversation by completing each subtask without having to handle multi-turn context. In short, the SHARDED Translation task simulates multi-turn conversations that are not underspecified.

We now list task properties we believe are important in leading models to get lost in conversation in multi-turn settings. First, generative tasks (*i.e.*, unlike extractive QA or classification) are more prone to model confusion, as they typically involve editing and refinement of new content. Second, the generative tasks should be sufficiently complex, involving multiple explicit specifications that will yield a multitude of shards. For example, an instruction: "Write a Python program that calculates $1 + 1$" is too simple to shard. Third, the solution or answer should be non-decomposable, such that revealing a shard modifies the entire solution (unlike the translation task, where each additional shard only asks to translate and append to the ongoing solution). We hypothesize that LLMs tested on tasks with the aforementioned three properties will likely get lost in conversation, evidenced by a large drop in averaged performance and reliability in SHARDED simulations.

7.4 Implications for Users of Conversational Systems

Users of LLM-based products should be aware of the lack of reliability of LLMs, particularly when used in multi-turn settings. Generally available generative technology is new, and prior work has identified the randomness in LLM-generated text as a point of confusion for users [55, 81, 77, 43]. We make two practical recommendations that can help users of LLM-based systems get the most out of their exchanges.

If time allows, try again. If a conversation with an LLM did not lead to expected outcomes, starting a new conversation that repeats the same information might yield significantly better outcomes than continuing an ongoing conversation. This is because current LLMs can get lost in the conversation, and our experiments show that persisting in a conversation with the model is ineffective. In addition, since LLMs generate text with randomness, a new conversation may lead to improved outcomes.

Consolidate before retrying. Since LLMs are ineffective at dealing with information dispersed across multiple turns, consolidating instruction requirements into a single instruction is an effective strategy to improve the model's aptitude and reliability (as shown by the CONCAT experiments). When a user notices that a model is lost in conversation, they can ask the LLM: "Please consolidate everything I've told you so far," then bring the response to a new conversation, alleviating the need for manual consolidation. In practice, there is anecdotal evidence that early adopters of LLM-based applications are aware that LLMs get lost in conversation. For example, users of the Cursor LLM-based coding environment report that frequently creating new conversations "whenever they can" is a recommended strategy to ensure high quality responses even though the tool allows to keep conversations going indefinitely.[3]

These two recommendations remain cumbersome for users and can only offer patched solutions rather than a principled approach. Once future LLMs can more reliably handle multi-turn conversations, the need for such recommendations should be alleviated, allowing users to communicate underspecified instructions over multiple turns naturally with less risk of the model getting lost in conversation.

[3] https://www.reddit.com/r/cursor/comments/1j72r8d/when_to_start_a_new_chat/