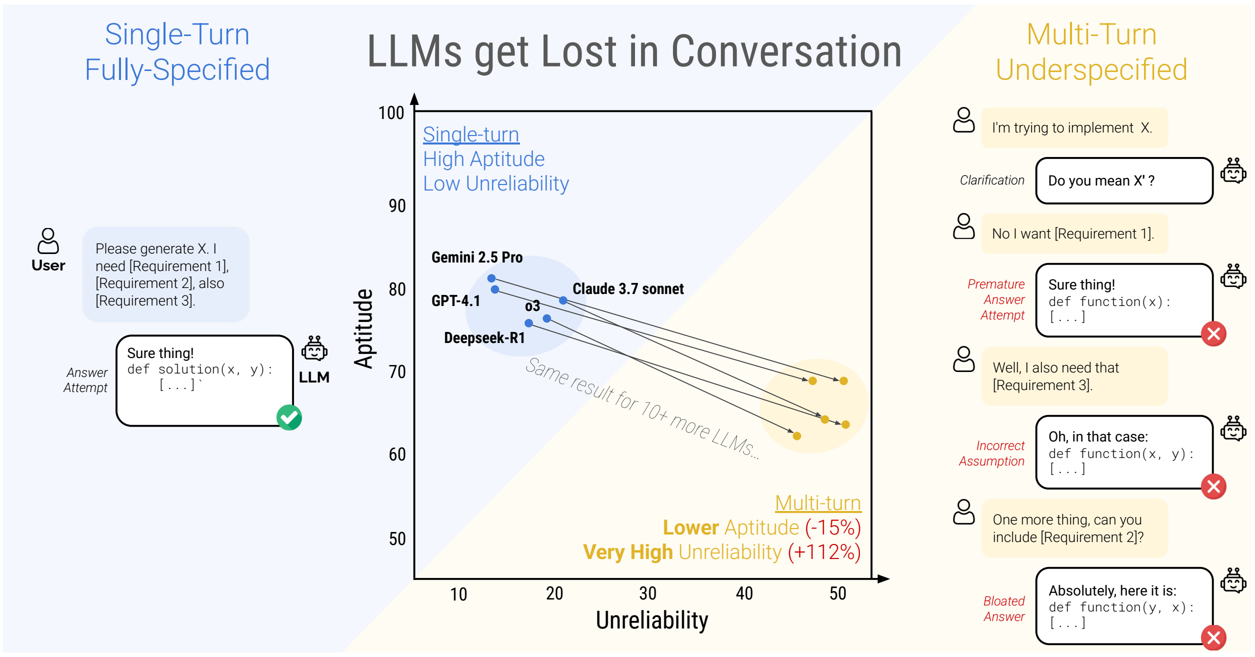
LLMs Get Lost In Multi-Turn Conversation

Philippe Laban[*][◇] **Hiroaki Hayashi**[*][♣] **Yingbo Zhou**[♣] **Jennifer Neville**[◇]
[◇]Microsoft Research [♣]Salesforce Research
{plaban,jenneville}@microsoft.com
{hiroakihayashi,yingbo.zhou}@salesforce.com

Abstract

Large Language Models (LLMs) are conversational interfaces. As such, LLMs have the potential to assist their users not only when they can fully specify the task at hand, but also to help them define, explore, and refine what they need through multi-turn conversational exchange. Although analysis of LLM conversation logs has confirmed that underspecification occurs frequently in user instructions, LLM evaluation has predominantly focused on the single-turn, fully-specified instruction setting. In this work, we perform large-scale simulation experiments to compare LLM performance in single- and multi-turn settings. Our experiments confirm that all the top open- and closed-weight LLMs we test exhibit significantly lower performance in multi-turn conversations than single-turn, with an average drop of 39% across six generation tasks. Analysis of 200,000+ simulated conversations decomposes the performance degradation into two components: a minor loss in aptitude and a significant increase in unreliability. We find that LLMs often make assumptions in early turns and prematurely attempt to generate final solutions, on which they overly rely. In simpler terms, we discover that **when LLMs take a wrong turn in a conversation, they get lost and do not recover**.

`Microsoft/lost_in_conversation` 🤗 `datasets/Microsoft/lost_in_conversation`



Please generate X. I need [Requirement 1], [Requirement 2], also [Requirement 3].

Sure thing!
```
def solution(x, y):
    [...]`
```

I'm trying to implement X.

Do you mean X' ?

No I want [Requirement 1].

Sure thing!
```
def function(x):
[...]
```

Well, I also need that [Requirement 3].

Oh, in that case:
```
def function(x, y):
[...]
```

One more thing, can you include [Requirement 2]?

Absolutely, here it is:
```
def function(y, x):
[...]
```

Figure 1: In this work, we simulate single- and multi-turn conversations for six generation tasks. The 15 LLMs we test perform much worse in multi-turn settings (-35%) explained by some loss in aptitude, and large losses in reliability. Aptitude is defined as performance in best-case conversation simulation, and unreliability as the gap between best- and worst-case performance. In short, we find that LLMs get lost in multi-turn, underspecified conversation.

[*]Equal Contributions

1 Introduction

Today's large language models (LLMs) function as conversational interfaces (*e.g.*, ChatGPT, Gemini, Claude), enabling users to interact with the LLM through multiple conversation turns. Such interaction promises to help users not only when they know what they need (i.e., they can fully specify their requirements in an instruction), but also when they don't. In such cases, users might start with an underspecified instruction and further clarify their needs through turn interactions. Though studies of LLM conversation logs have confirmed that underspecification in user instructions is prevalent [27], LLM systems are typically evaluated in single-turn, fully-specified settings.

Even though a growing body of work proposes to evaluate LLMs in a **multi-turn** fashion, we identify in our review (Section 2) that most prior work treats the conversation as *episodic*: conversation turns might relate to each other, but the conversation can effectively be decomposed as an array of subtasks that can be evaluated in isolation. We argue that episodic tasks move away from what is prevalent in human conversation: underspecification [91, 27].

In this work, we close this gap by creating a simulation environment for multi-turn underspecified conversations – sharded simulation – that leverages existing instructions from high-quality single-turn benchmarks. At a high level, the sharding process we propose transforms existing single-turn instructions into *sharded instructions*, a set of smaller instructions that jointly deliver the same information as the original instruction. Sharded simulation then ensures that each turn of conversation reveals at most one shard of information per conversation turn, enforcing that the instruction is gradually revealed through the conversation.

On the set of tasks that we experimented on, we observed that models engaged in multi-turn underspecified conversations achieved an average performance of 65%–a 25-point drop from single-turn performances of 90% when they receive the entire instruction at the beginning of the conversation. Notably, we observe this drop in performance even in two-turn conversations, and across all LLMs we test, from small open-weights (LLama3.1-8B-Instruct) to state-of-the-art (Gemini 2.5 Pro).

Furthermore, we decompose the performance degradation into two components: (1) loss in aptitude, and (2) increase in unreliability. We find that in single-turn settings, models with higher aptitude tend to be more reliable (*e.g.*, GPT-4.1, Gemini 2.5 Pro). On the other hand, all LLMs exhibit very high unreliability in multi-turn settings, regardless of aptitude. We refer to this as the *lost in conversation phenomenon*: when LLMs take a wrong turn in multi-turn conversation, they get lost and do not recover.

We investigate several explanations for this effect and show that the LLMs tend to (1) generate overly verbose responses, leading them to (2) propose final solutions prematurely in conversation, (3) make incorrect assumptions about underspecified details, and (4) rely too heavily on previous (incorrect) answer attempts.

Our findings highlight a gap between how LLMs are used in practice and how the models are being evaluated. Ubiquitous performance degradation over multi-turn interactions is likely a reason for low uptake of AI systems [73, 4, 28], particularly with novice users who are less skilled at providing complete, detailed instructions from the onset of conversation [87, 35].

The rest of the paper is structured as follows: Section 2 situates our work with respect to prior work on multi-turn evaluation. In Section 3, we describe the simulation environment we built for both single- and multi-turn conversations on a diverse set of generation tasks. We introduce the six tasks and the metrics we use to evaluate the aptitude and reliability of models in Section 4.1. Sections 5-6 define our main experiment involving 15 LLMs, and analyze the main findings. Finally, the Implications section (Section 7) discusses the ramifications of the work, from the perspective of organizations that are building LLM-based conversation products, to that of end-users of the LLM-based systems. We provide actionable recommendations based on small-scale experiments and make a concrete call-to-action to LLM builders, urging them to prioritize multi-turn reliability in conjunction with aptitude in future model iterations.

2 Background and Related Work

Previous-generation language models (e.g., BART [45], GPT-2 [65], or T5 [66]) were not equipped to handle multi-turn conversations, which led evaluation to focus on single-turn tasks [79]. Conversational AI was typically implemented as specialized systems that leveraged language models as components [36], and were evaluated through human protocols [17, 42, 21, 54], or competitions like Amazon's Alex Prize [67].

As the meteoric rise of ChatGPT led to increased interest in multi-turn evaluation, initial popular efforts such as MT-bench [89] leveraged crowd-sourced annotations to evaluate LLM-as-a-judge ability. Follow-up works expanded on MT-bench, for instance to include longer conversations [37, 18], increase evaluation granularity [2], or to tackle different aspects such as naturalness [72] or tool use [85, 80].

Crucially, such works typically simulate *episodic* conversations: each turn in the conversation introduces a subtask that relates to previous conversation turns, but can be evaluated in isolation. In this work, we find that episodic tasks overestimate LLM performance in multi-turn conversations (see Section 7.3). In short, although episodic tasks require some level of multi-turn context understanding, they do not involve actively fusing the information to answer *underspecified* user instructions. Underspecified user instructions are not only common in real-world human-AI communication [27], but also a natural tendency in conversations, termed "the principle of least effort" [91]. We show that underspecification in multi-turn conversations leads to large and universal performance degradations: LLMs make early assumptions to fill in for missing information, prematurely attempt to propose finalized solutions, and have difficulty adapting and course-correcting when provided with new information. We make underspecification the central element of our evaluation setting.

Multi-turn episodic evaluation is sometimes framed as a way to evaluate multi-turn model capabilities with higher granularity. Categories of subtasks (such as refinement, follow-up, expansion, etc.) allow for the study of more specific LLM behavior [2, 37, 74, 19, 16, 48, 25]. According to such framing, multi-turn tasks *differ* from single-turn tasks and are not evaluated on the same set of tasks. We argue that this framing is artificial and limits the scope of multi-turn evaluation, restricting the direct comparison of multi-turn and single-turn abilities of LLMs. In our work, we conduct both single-turn and multi-turn conversation simulations on *a common set of tasks*: controlled experiments that precisely allow us to identify performance degradations from single- to multi-turn settings.

Evaluating LLMs in multi-turn settings is a challenge because conversational trajectories diverge far more than in a single-turn. Thus, most previous studies have focused on classification or short-form tasks, with more straightforward evaluation settings. However, the predominant use cases for LLMs are generative in nature, both for programming (*e.g.*, coding assistants) and natural language (*e.g.*, writing, summarizing) [88, 26]. Long-form evaluation in the multi-turn setting is therefore essential, as it assesses models' ability to flexibly adapt and refine the response as the users provide more information. In this work, we focus exclusively on generation tasks that capture widely used scenarios in both programming and natural language domains.

Scaling multi-turn experimentation requires simulating a user. Existing studies implemented such user simulation in different ways: relying on templates [12, 68, 39, 16], using an LLM [63, 46, 7, 48], involving human annotators [21, 7], or real users as part of a study [67, 38, 11]. Although involving real users leads to the most natural and realistic conversations, it comes at the cost of scalability and reproducibility. In this work, we adopt an LLM-based simulator to enable controlled flexibility and divergence. Nevertheless, a fully automated simulation limits the scope of our findings: the conversations we simulate are not representative of human-AI conversations. We therefore frame the simulation as a tool to study the *LLM behavior* in the multi-turn setting rather than user behavior. In addition, as detailed in the Limitations Section (Section 9), we argue that our simulation framework is simplistic and idealized. For example, the conversations are guaranteed to end with sufficient information to solve the tasks, and the simulator limits unexpected behavior (*e.g.*, derailing) that can occur in real-world settings. We suggest these choices imply that degradations observed in this work are most likely underestimates of what occurs in real-world, underspecified multi-turn Human-AI conversations. Appendix A introduces other related work specifically focused on underspecified communication.

3 Simulating Underspecified, Multi-Turn Conversation

To assess LLM performance in multi-turn, underspecified conversation, we develop a simulation environment that repurposes existing tasks from single-turn benchmarks. First, we apply a *sharding process* to transform original fully-specified instructions into *sharded instructions*. Second, we implement a sharding simulation environment that carries out a multi-turn conversation based on a sharded instruction.

3.1 Sharding Process: From Fully-Specified to Sharded Instructions

An original, fully-specified instruction from GSM8K [14] and the equivalent sharded instruction are listed in Figure 2.

The original instruction is a single, long utterance that introduces all the content at once: a high-level question (*i.e.*, "How long will it take [...]"), context, and conditions. The sharded instruction is composed of *a set of shards*, each introducing a single element from the original instruction. More specifically, the first shard (Shard 1) of a sharded instruction always introduces the high-level intent for the instruction, and subsequent shards each provide clarification to the instruction. Taken jointly, the set of shards reflects the same information provided in the fully-specified instruction, with the information explicitly divided across shards.

In Appendix B, we provide a more precise and mathematical definition of a sharded instruction in relation to the original fully-specified instruction, and define five key properties a sharded instruction must satisfy to be considered valid.