table with the highlighting present, the third shard provides the Wikipedia page name, the fourth shard provides the Wikipedia Section name. Finally, a fifth shard provides a fixed set of 10 randomly-selected example captions from the training set of the ToTTo dataset.

Each instruction in ToTTo is assigned one to three reference captions that were collected by authors of the original dataset. Evaluation on a candidate caption calculates the BLEU score [58] between the candidate and the set of available references, following the evaluation methodology from the original paper.

The Data-to-Text is a refinement task; at each turn, the model is provided an additional shard of information, and is explicitly told to update its response considering all the information provided so far. As a refinement task, assistant responses at each turn are automatically categorized as answer attempts, and the extracted answer is considered to be the entire response. The system instruction informs the model that its response should consist solely of a table caption, without additional text (such as intro, outro, or politeness wording).

I.6 Summary

The Summary instructions are based on samples of the Summary of a Haystack dataset [40]. We reuse the entire instructions from Summary of a Haystack to produce 92 sharded instructions. The original instructions each consist of a *haystack* – 100 documents for a total of 100,000 tokens of content – and a user query. The goal of the task is to generate a bullet-point-formatted summary of the query-relevant insights that occur in the collection of documents, and use citation to attribute information in each of the bullet points back to the source documents.

The original setting of the Summary of a Haystack purposefully includes a large amount of redundancy (each insight is repeated across at least 6 documents) to evaluate LLMs' ability to thoroughly cite sources. However, we simplify the task for the multi-turn setting, as the 100,000-token haystacks restrict the variety of models we can evaluate. We instead follow subsequent work in selecting smaller Haystacks ("mini-Haystacks") [3]. Mini-Haystacks consist of 20 documents and ensure that each reference insight is repeated across three documents. For each instruction, we produce ten shards by randomly assigning two documents per shard. The initial shard further specifies high-level task instruction, by specifying the user query, the expected bullet-point format, with a formatted citation.

Summary of a Haystack relies on an LLM-based metric (Joint Score) to compute the quality of the summary in terms of both the relevance of the candidate bullet points (coverage) and the quality of the generated attribution within the bullet points (citation). The authors note that the metric is recall-based, such that longer summaries are likely to score higher than shorter ones. To account for length bias, the original task instructs models to generate summaries of at most 300 words, which we include in our experiments as well. Specifically, models are instructed in all settings to generate summaries of up to 300 words. We observed that in multi-turn settings, models often *forget* this instruction, leading to non-adherence to the instruction. To avoid penalizing models that correctly remain within the 300-word limit, we truncate summaries that go beyond the limit, removing words in equal proportion from summary bullet points, such that evaluated summaries all respect the 300-word limit. We note that this tendency for LLMs to go beyond is further discussed in Appendix F, where we observe that across tasks, model answer attempts get "bloated" over turns of conversations. In single-turn settings (full, concat), LLMs largely respect the 300-word length limit.

The summary task is a refinement task. Assistant responses at each turn are automatically categorized as answer attempts, and the entire response is considered to be the extracted answer.

I.7 Translation

The Translation instructions were collected from the WMT 2019 task on document-level translation [70]. Specifically, we selected 30 documents German-English pairs. Document pairs are aligned at the sentence level (i.e., English and German documents in a pair have the same number of sentences). We truncated the selected pairs to their first ten sentences, and sharded the document instruction such that each shard would introduce exactly two sentences from the document, for a total of five shards. We provided shards in German, and the task consisted in translating into English (i.e., German$\rightarrow$English). Hence, Shard 1 introduces the first two German sentences, Shard 2 introduces German sentences 3-4, etc. In the sharded setting, the task requires the LLM to translate the document with all the provided sentences so far. In the full settings, the LLM is provided the entire document (10 sentences) in the first turn. In the concat setting, the LLM is also provided all sentences in the first turn, but separated into the shards (two sentences at a time).

In initial experiments, we experimented with other sharding strategies, including breaking shards at a specific number of words (rather than sentence boundary), and increasing the length of documents (from 10 to 20 sentences), without observing significant differences in results. This led us to adopt the setting we describe: sharding every two sentences, and truncating at 10-sentences.

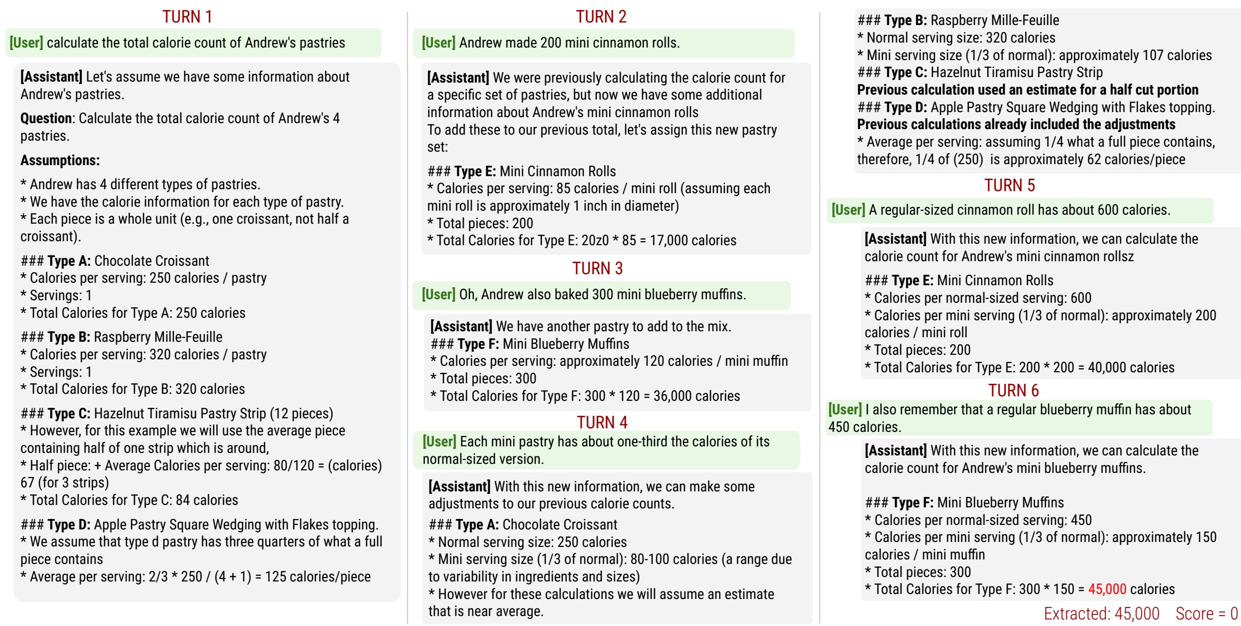

Figure 11: Example simulated multi-turn conversation for the Math task. This conversation simulation was with assistant model Llama3.1-8B-Instruct. The sharded instruction consists of six shards. The correct answer to the instruction is 85,000 calories.

We evaluated performance with the BLEU metric[58], the standard metric for translation tasks, which was used as well in the original WMT 2019 competition.

Appendix J Example Simulated Conversation

Figure 11 provides an example conversation that was simulated during our experiments in the sharded setting. The simulation was conducted on the Math task, with a 6-shard instruction, and using the Llama3.1-8B-Instruct as the assistant. This conversation illustrates the following properties described in the rest of the paper: (1) the LLM makes assumptions early in the conversation (in Turn 1, describing four pastries that are irrelevant), (2) although it correctly interprets user-provided information, it also unnecessarily updates the information for assumptions it made (Turn 4), (3) this leads to unnecessary complexity, and the model ultimately forgets that the initial instruction was to calculate total calorie count, and returns only half of the calculation (just for Mini Blueberry Muffin). In short, this conversation illustrates the lost in conversation phenomenon: when the user instruction is underspecified (Turns 1-4), the LLM makes assumptions that detract from the conversation and lead to incorrect or incomplete answers.

Appendix K Gradual Sharding Implementation

To evaluate the effect of instruction granularity on performance degradations, we conducted the *gradual sharding experiment*.

We selected sharded instructions that had exactly eight shards, leading to a total of eight instructions across three tasks (Code, Math, Data-to-Text). We then leveraged an LLM (GPT-4o) to expand each instruction into 7 variants with differing number of shards. The LLM was instructed to *merge* the original sharded instruction into a smaller sharded instruction with two to seven shards. The instruction authorized minor rephrasing to allow for individual shards to be fluent, but encouraged the LLM to remain as close as possible to the original instruction in wording.

As such, each of the original instruction can be paired to: (1) a concat instruction (one-shard), and (2) 7 sharded instructions, ranging from two to eight shards. Applying this method to the 31 instructions yields a total of 248 instructions, with an equal number for the number of shards (from 1 to 8) and on the identical underlying problems.

We ran simulations using the 248 instructions, simulating 10 conversations per instruction and model for two models: GPT-4o and GPT-4o-mini. Findings of the gradual sharding experiment are described in Section 6.3.

Appendix L Temperature Experiment Implementation

To evaluate the effect of temperature on aptitude and reliability of LLMs in single- and multi-turn settings, we conducted the following *temperature experiment*.

We selected 10 instructions from each of four tasks: Code, Database, Actions, and Math (for a total of 40). We ran experiments with two models (GPT-4o and GPT-4o-mini). For each instruction and each temperature combination, we conducted simulations for three conversation settings: full, concat, and sharded. For each conversation setting, we varied temperature parameters to three values: 0.0, 0.5, and 1.0. For the full and concat setings, this corresponds to three temperature combinations (as only the assistant temperature can be modified), whereas there are a total of nine combinations for the sharded setting, as both the assistant and user temperature is varied.

We chose to increase the number of simulations to 20 runs per condition (compared to 10 in the main experiment), as the focus of the experiment is to measure variations in model aptitude and reliability, and added simulation runs lead to better percentile estimates used in calculating metrics. This added requirement was not computationally expensive as the temperature experiment involved a limited number of models (2 vs. 15) and instructions (40 vs. 600) in comparison to our main experiment.

Findings of the experiments are described in Section 7.2.

Appendix M Recap & Snowball Experiment Implementation

We leverage SHARDED conversation logs to simulate RECAP setting, since RECAP only differs from SHARDED in terms of an additional recapitulation turn that gathers all the previous user utterances. This implementation also allows us to directly compare the effect of the approach against the SHARDED results. Specifically, for each SHARDED simulation run, we appended the "recap" turn and run the simulation one more turn. Since it requires stacking the past turns every turn, we simulate the entire conversations from scratch for SNOWBALL simulations. The prompt concatenates the previous turn user utterances as bullet points, followed by the text for the current turn: `Just to reiterate:\n - [past utterance 1]\n- [past utterance 2]\n\n Also,\n[current utterance]`. We note that what is accumulated for both RECAP and SNOWBALL are verbalized utterances from the user simulator, not the original shards themselves. For both simulation settings, we run $N = 10$ simulations on all of the sharded instructions on four tasks (Code, Database, Math, Actions) and report the mean of averaged performance over the tasks, which is shown in Table 2.

Appendix N On obtaining deterministic outputs from LLMs

As we demonstrated in our experimental results, setting the temperatures to zero still leads to high unreliability, due to compounding effect of subtle non-determinism over tokens and turns.

In theory, greedy decoding (*i.e.*, $T = 0$) will always pick the argmax over the vocabulary distribution. However, it is reported that hardware limitations on floating point operations cause slightly different intermediate values, which results in a ripple effect of larger value changes and therefore different tokens being selected.

Notable model providers acknowledge the non-determinism implicitly or explicitly; Anthropic recommends sampling multiple times to cross-validate output consistency,[4] Google also highlights that their model outputs are *mostly* deterministic,[5] and OpenAI recommends setting `seed` parameter to further reduce the non-determinism.[6]

Nevertheless, we caution users that multi-turn conversations can be increasingly unreliable owing to divergent LLM responses.

[4] https://docs.anthropic.com/en/docs/test-and-evaluate/strengthen-guardrails/reduce-hallucinations.

[5] https://cloud.google.com/vertex-ai/generative-ai/docs/learn/prompts/adjust-parameter-values#temperature.

[6] https://platform.openai.com/docs/advanced-usage#reproducible-outputs.