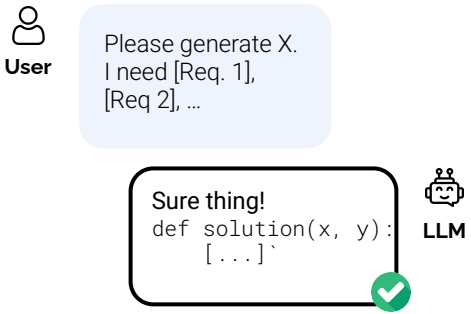
LLMs Get Lost in Multi-turn Conversation

Philippe Laban* **Hiroaki Hayashi*** Yingbo Zhou Jennifer Neville

Microsoft Research and Salesforce Research

8/26/2026
Presented @ ICLR 2026

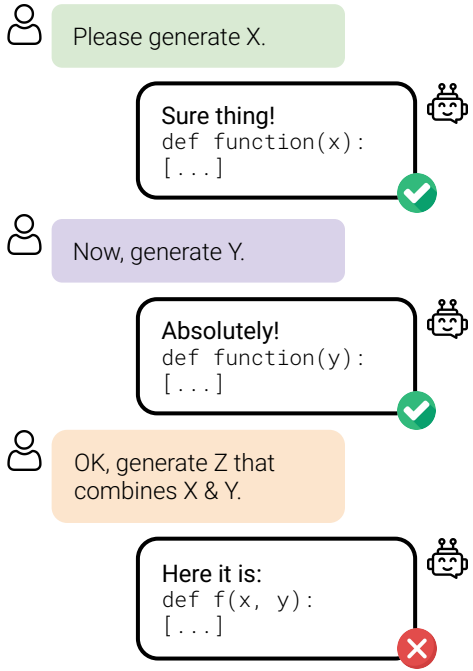
LLM are *conversational*, so should the evaluation be



シングルターン型指示追従評価データ

- LLM評価に広く使われる
- 指示内に情報の過不足はない

LLM are *conversational*, so should the evaluation be



既存の**マルチ**ターン型指示追従評価データ

- LLMの用途 (e.g. ChatGPT) に近い設定
- 基本的に***Episodic***:

  ターン間での文脈のつながりが薄い
- 文脈を考慮した評価ベンチマーク作成には人手が必要

課題：

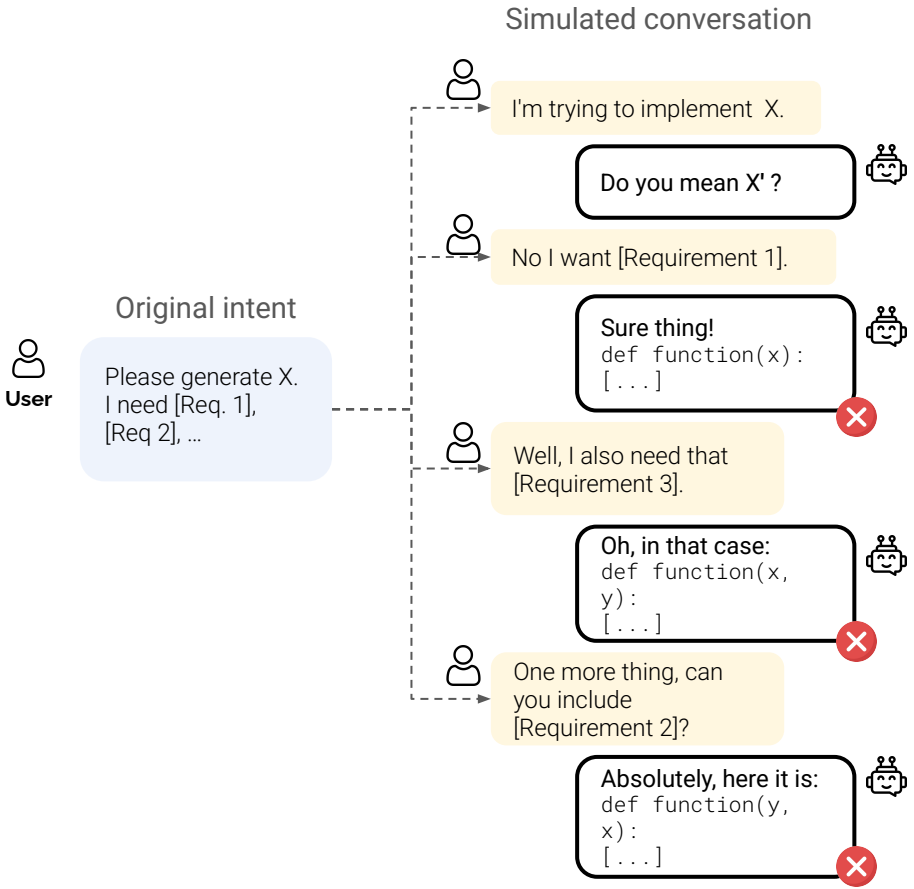
- LLMは複数ターンにまたがって表現される意図を理解できるか？
- シングル vsマルチターン間でLLMの性能はどう変わるか？

LLM are *conversational*, so should the evaluation be

本研究のゴール :

マルチターンかつ曖昧な状況下 での評価

- なるべく実用により近い会話ダイナミクスを実現する
- シングルターン評価データから作成することで、公平な性能比較を目指す



Back story

HumanEval (Pythonコーディングベンチマーク)

User

Please implement the following function:

```
from typing import List

def below_zero(operations: List[int]) -> bool:
    """You are given a list of deposit and
withdrawal operations on a bank account that
starts with zero balance. Your task is to
detect if at any point the balance of account
falls below zero, and at that point function
should return True. Otherwise it should return
False.
    >>> below_zero([1, 2, 3])
    False
    >>> below_zero([1, 2, -4, 5])
    True
    """
```

Hiro: LLM評価に使うには問題が単純すぎるし、こんなにしっかりスペックを準備しないので、現実的ではない。

Phil: 誰もLLMにこんな問い方はしないので、現実的ではない。

Back story

HumanEval

Please implement the following function:

```
from typing import List

def below_zero(operations: List[int]) -> bool:
    """You are given a list of deposit and
withdrawal operations on a bank account that
starts with zero balance. Your task is to
detect if at any point the balance of account
falls below zero, and at that point function
should return True. Otherwise it should return
False.
    >>> below_zero([1, 2, 3])
    False
    >>> below_zero([1, 2, -4, 5])
    True
    """
```

→

Sharded-HumanEval

Write the python function `below_zero` that finds out if account balance is ever negative

The input is a list of ints that represent transaction amounts

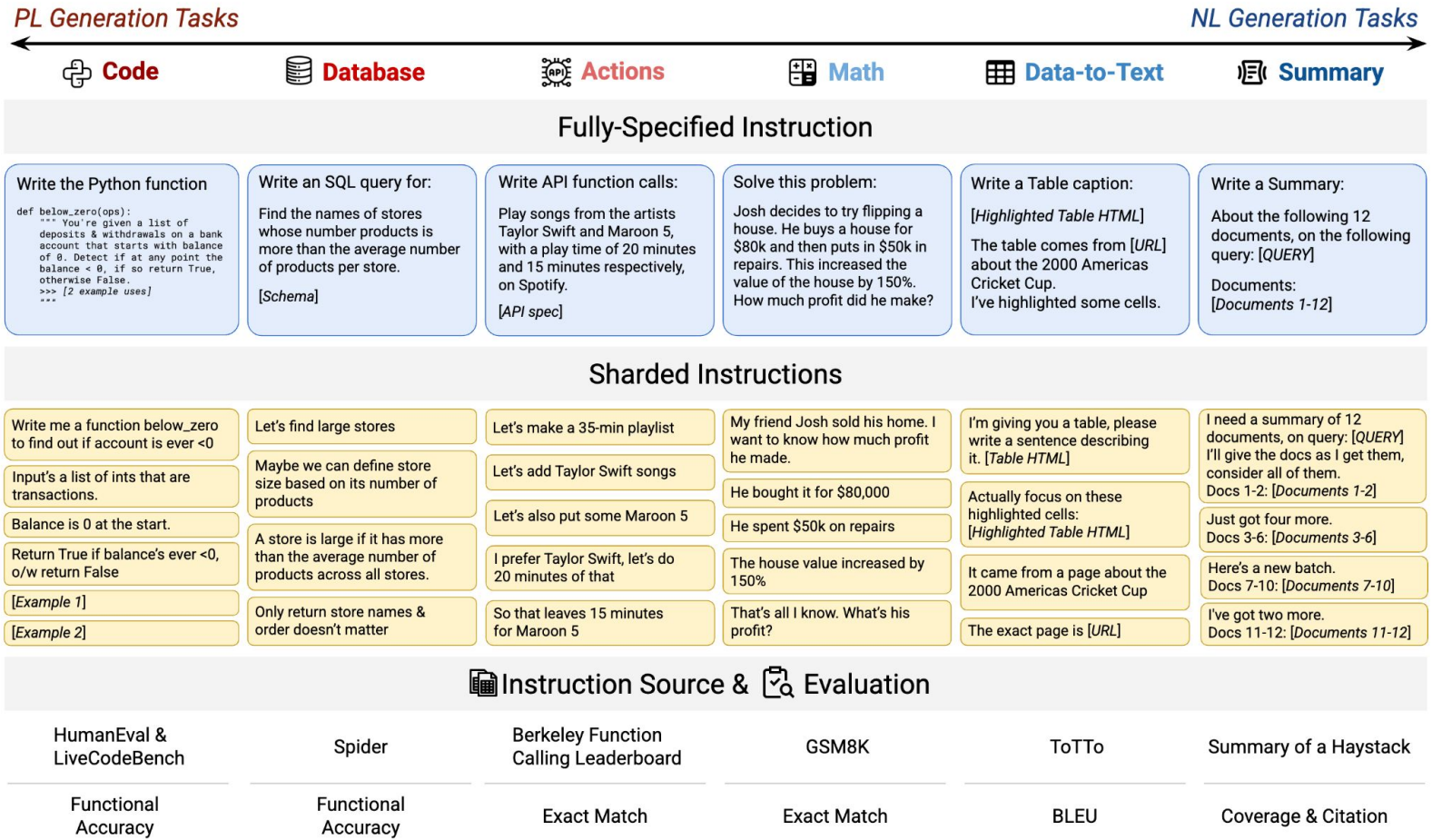
You can assume the initial balance is 0

Return True if the balance's ever negative, and False otherwise

Example 1

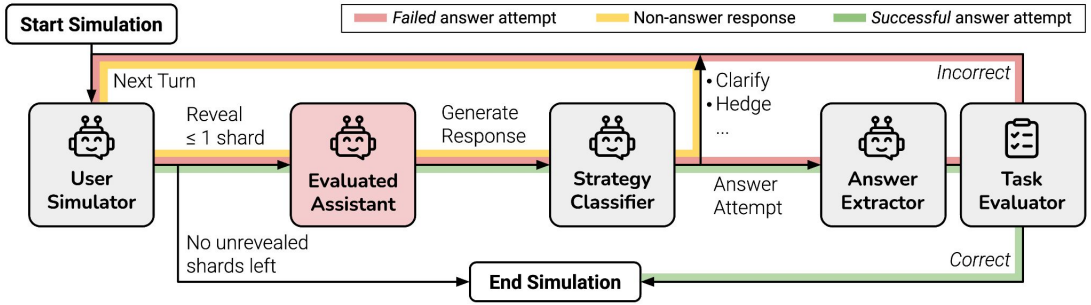
Example 2

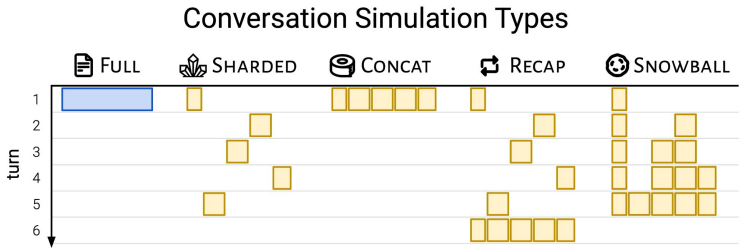
元のタスクを分解し、集合体として同じ指示となるような**shard**を作る


PL Generation Tasks
NL Generation Tasks
Code
Database
Actions
Math
Data-to-Text
Summary
Fully-Specified Instruction
Write the Python function
def below_zero(ops):
""" You're given a list of deposits & withdrawals on a bank account that starts with balance of 0. Detect if at any point the balance < 0, if so return True, otherwise False.
>>> [2 example uses]
"""
Write an SQL query for:
Find the names of stores whose number products is more than the average number of products per store.
[Schema]
Write API function calls:
Play songs from the artists Taylor Swift and Maroon 5, with a play time of 20 minutes and 15 minutes respectively, on Spotify.
[API spec]
Solve this problem:
Josh decides to try flipping a house. He buys a house for $80k and then puts in $50k in repairs. This increased the value of the house by 150%. How much profit did he make?
Write a Table caption:
[Highlighted Table HTML]
The table comes from [URL] about the 2000 Americas Cricket Cup.
I've highlighted some cells.
Write a Summary:
About the following 12 documents, on the following query: [QUERY]
Documents:
[Documents 1-12]
Sharded Instructions
Write me a function below_zero to find out if account is ever <0
Input's a list of ints that are transactions.
Balance is 0 at the start.
Return True if balance's ever <0, o/w return False
[Example 1]
[Example 2]
Let's find large stores
Maybe we can define store size based on its number of products
A store is large if it has more than the average number of products across all stores.
Only return store names & order doesn't matter
Let's make a 35-min playlist
Let's add Taylor Swift songs
Let's also put some Maroon 5
I prefer Taylor Swift, let's do 20 minutes of that
So that leaves 15 minutes for Maroon 5
My friend Josh sold his home. I want to know how much profit he made.
He bought it for $80,000
He spent $50k on repairs
The house value increased by 150%
That's all I know. What's his profit?
I'm giving you a table, please write a sentence describing it. [Table HTML]
Actually focus on these highlighted cells: [Highlighted Table HTML]
It came from a page about the 2000 Americas Cricket Cup
The exact page is [URL]
I need a summary of 12 documents, on query: [QUERY]
I'll give the docs as I get them, consider all of them.
Docs 1-2: [Documents 1-2]
Just got four more.
Docs 3-6: [Documents 3-6]
Here's a new batch.
Docs 7-10: [Documents 7-10]
I've got two more.
Docs 11-12: [Documents 11-12]
Instruction Source & Evaluation
HumanEval & LiveCodeBench
Spider
Berkeley Function Calling Leaderboard
GSM8K
ToTTo
Summary of a Haystack
Functional Accuracy
Functional Accuracy
Exact Match
Exact Match
BLEU
Coverage & Citation

Simulating the interactions with a LLM

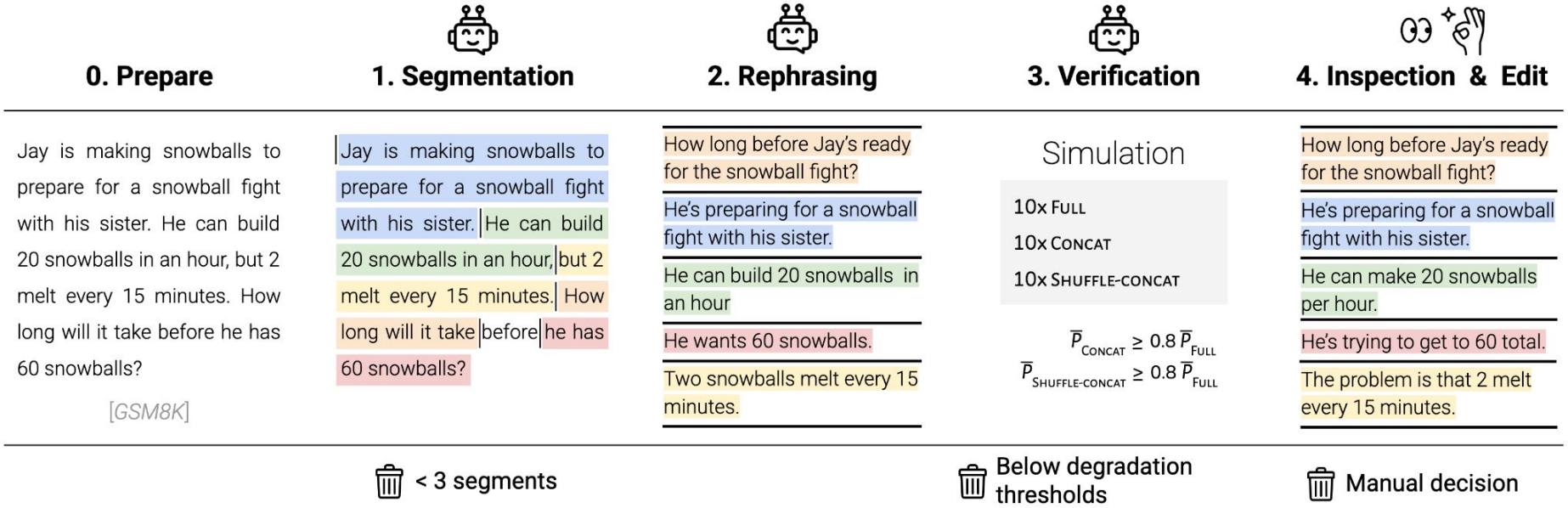
- ユーザー行動『徐々に意図のshardを補足』はLLMを使って模倣
  - e.g.,「そういえばこうだった」「ちなみにこれこれは特別に処理して」
  - 完全な人手によるアノテーションよりスケーラブル、かつ細かな制御が可能
  - LLMによりshardを自然な応答に織り込むことが可能
- 評価対象のLLM (assistant) に対して、様々な方法でshardを明らかにする
  - Full / Sharded
  - **Concat**: 全てのshardを一度に指示
  - **Recap**: 最後のもう1ターンで全てのshardを同時に指示 :「これまでの指示をまとめると...」
  - **Snowball**: 雪崩式に過去のshardに加えて指示 :「今までの指示を踏まえて、これをこうして」





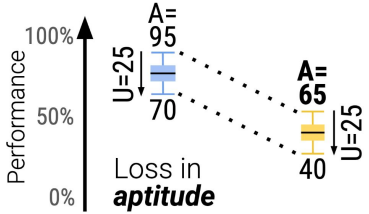
Transforming single-turn tasks into **shards**

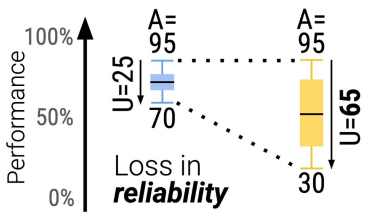
- 情報の過不足なくシングルターンのタスクを分解
  - 明らかな外れ値のみを手動で検閲
- Shuffle-concat によりshardの順不同性を担保

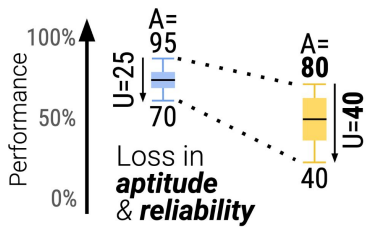


Experiments

- LLMs: **オープン**, **クローズドなどたくさん**
  - Small to large (Llama 7B, Gemini, Claude, ...)
  - Non-reasoning / Reasoning models (Deepseek R1, o3, ...)
- **生成タスク**
  - Python, SQL**生成**, **関数呼び出し**, **算数**, Data-to-Text, **要約**
  - **各タスク**~100**サンプル**, **シミュレーション** x10
- Evaluation
  - **平均性能** (Performance): **平均的なパフォーマンス**
  - **潜在性能** (Aptitude): **最良時の性能**（90**パーセンタイル**）
  - **不安定性** (Unreliability): **最良時と最悪時の性能差**（90**パーセンタイル** − 10**パーセンタイル**）

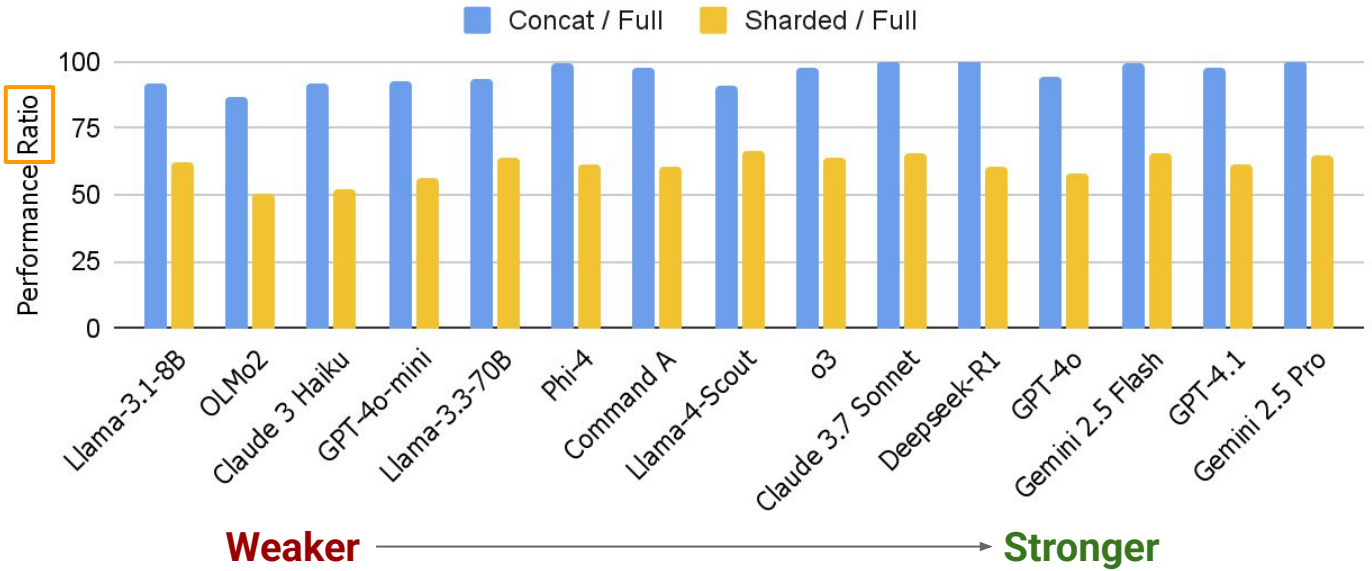






Averaged Performance drops *universally* by 39%

- **すべてのモデルにおいて** 平均性能は同様に落ちる (-39%)
  - 推論モデルが特別頑強であるというわけではない
  - 性能の低いモデルほど、Concat自体の性能も下がる

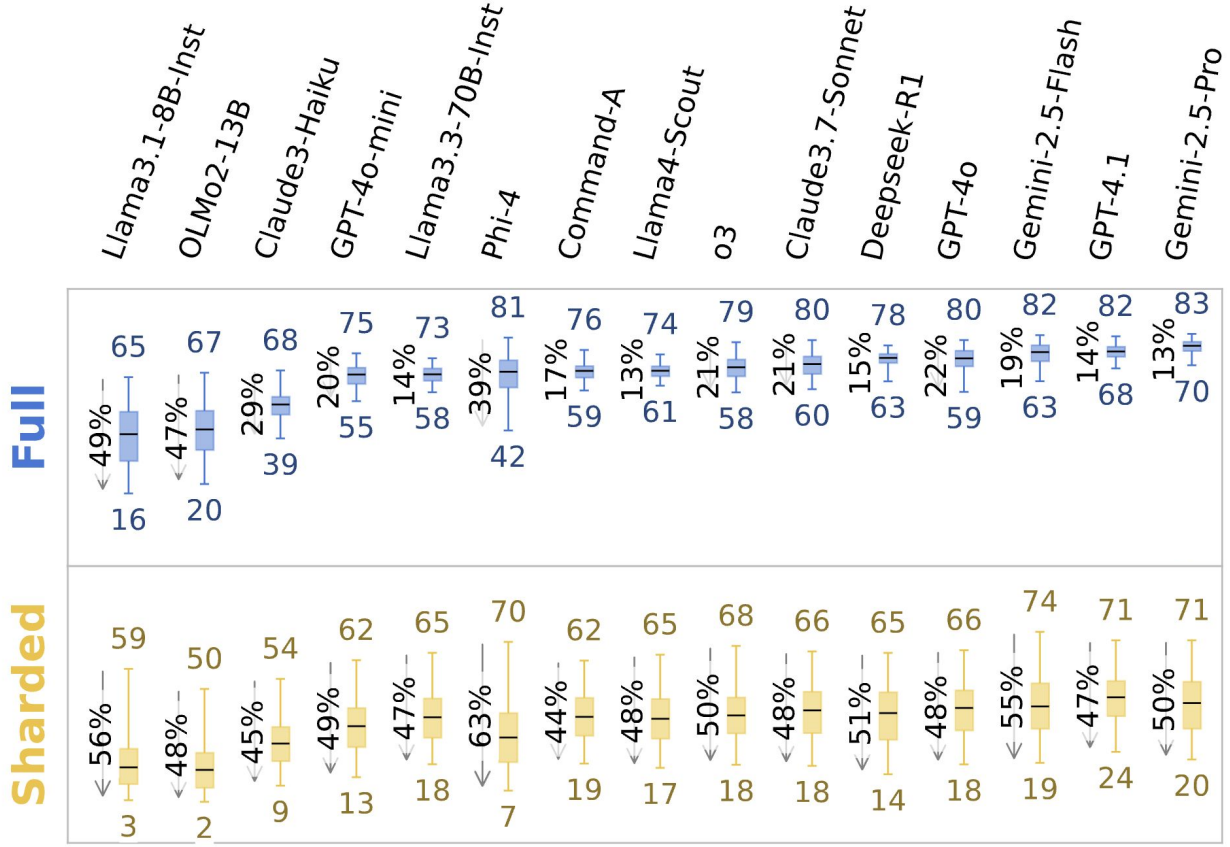


新しいモデルも同様の性能落差

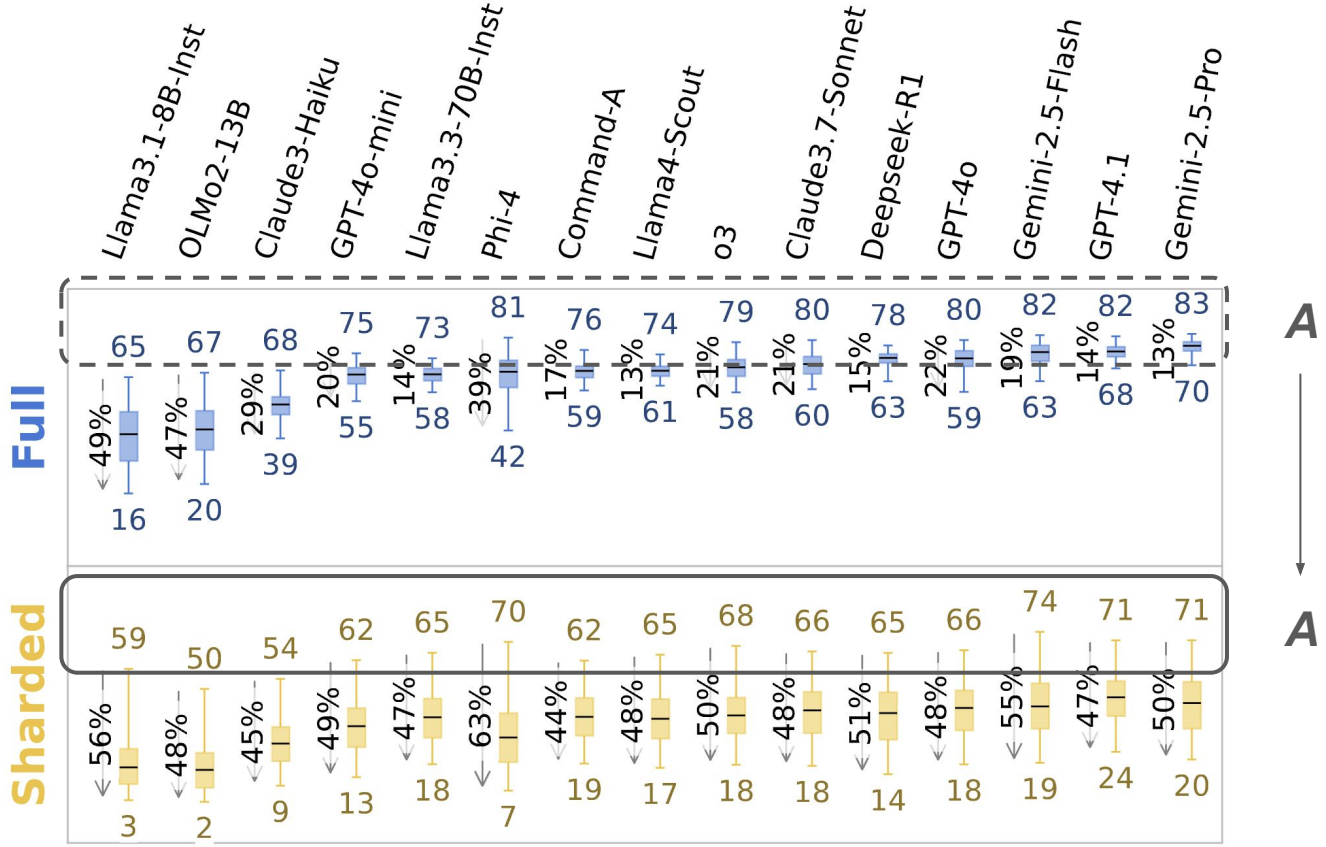
Model	FULL						CONCAT						SHARDED						Overall	
3.1-8B	27.4	64.1	82.9	13.7	63.9	7.6	21.2	47.7	83.0	15.8	62.6	6.5	21.7	25.9	45.5	13.3	37.4	3.4	91.6	62.5
OLMo2	18.8	54.8	56.1	17.2	80.0	-	16.3	40.5	49.8	14.3	80.1	-	14.4	22.4	13.8	9.1	46.3	-	86.5	50.5
Phi-4	53.2	87.6	82.7	24.0	89.2	-	48.4	79.6	76.0	28.7	90.4	-	39.1	33.1	34.1	23.2	52.5	-	99.0	61.7
4-Scout	73.9	92.7	98.0	35.3	96.3	13.7	60.3	81.5	98.3	28.3	92.9	13.7	46.4	27.1	69.9	26.1	67.0	12.3	91.0	66.0
CMD-a	72.0	91.9	98.5	34.3	94.5	24.3	61.6	86.1	98.4	37.9	91.9	21.3	44.9	33.6	72.0	30.1	66.0	4.9	95.7	58.3
o3	86.4	92.0	89.8	40.2	81.6	30.7	87.2	83.3	91.5	39.4	80.0	30.4	53.0	35.4	60.2	21.7	63.1	26.5	98.1	64.1
3.7-Sonnet	78.0	93.9	95.4	45.6	85.4	29.3	76.2	81.5	96.0	53.3	87.2	28.9	65.6	34.9	33.3	35.1	70.0	23.6	100.4	65.9
3.3-70B	72.0	91.1	95.0	-	91.7	15.8	52.7	87.9	97.0	-	91.8	14.7	51.6	35.4	71.0	-	61.5	10.5	93.1	63.8
4o	88.4	93.6	96.1	44.2	93.8	23.9	82.9	91.7	97.1	31.8	91.9	23.9	61.3	42.3	65.0	20.5	67.9	10.6	93.8	57.5
R1	99.4	92.1	97.0	30.7	95.5	26.1	97.1	89.9	97.0	38.9	92.9	24.4	70.9	31.5	47.5	22.0	67.3	17.2	102.1	60.4
4.1	96.6	93.0	94.7	54.6	91.7	26.5	88.7	86.5	98.5	54.4	89.7	26.8	72.6	46.0	62.9	28.6	70.7	13.3	97.9	61.8
2.5-Flash	97.0	96.3	88.4	52.2	90.6	29.1	92.5	95.5	89.2	52.0	88.4	29.4	68.3	51.3	42.6	31.0	66.1	26.1	99.0	65.6
2.5-Pro	97.4	97.3	97.8	55.0	90.2	31.2	95.7	94.9	98.1	57.1	89.3	31.8	68.1	43.8	36.3	46.3	64.3	24.9	100.1	64.5
5.2	87.3	94.0	91.2	38.0	93.9	33.6	90.9	81.0	90.4	37.0	88.3	32.8	74.1	38.5	34.3	27.7	72.0	30.2	96.4	67.2
5.1	84.5	93.2	91.0	50.8	92.7	30.2	88.1	80.4	90.9	41.1	92.9	30.2	33.4	37.2	59.9	30.1	71.5	23.6	95.3	60.0
5	91.3	89.0	85.8	50.8	94.3	34.2	91.3	75.8	87.7	42.5	91.2	33.7	30.1	31.2	51.6	23.8	67.2	30.8	94.4	56.0
5-chat	98.7	96.8	96.7	56.9	92.1	31.4	92.4	92.3	98.6	56.8	89.0	30.9	77.0	38.0	81.5	36.1	65.3	23.4	97.7	68.4
Grok-4	97.5	86.6	89.3	56.4	85.8	32.2	94.8	79.7	89.5	55.8	86.1	32.5	82.0	33.6	46.8	31.4	57.8	27.5	98.3	63.9
4.6-Sonnet	89.8	96.3	91.6	44.0	91.7	34.7	93.2	85.8	82.3	55.6	93.0	34.1	76.2	37.4	52.0	47.4	69.1	20.5	101.5	70.5
4.6-opus	100.0	96.3	94.5	47.3	93.0	37.1	99.2	92.5	94.5	56.7	88.7	36.6	86.8	48.6	41.5	51.2	72.2	14.6	101.6	67.8
3-Pro	100.0	97.6	98.5	57.1	94.2	33.4	98.8	98.3	97.1	61.2	90.9	33.1	89.6	63.4	54.7	52.9	73.4	29.6	100.2	78.2

Table 1: Averaged Performance ($\overline{P}$) of LLMs on six tasks (Code, Database, Actions, Data-to-text, Math, and Summary). For each task, conversations are simulated in three settings: FULL, CONCAT, and SHARDED. Background color indicates the degradation from the FULL setting. The last two columns average the performance drops from the CONCAT and SHARDED compared to the FULL in percentages across the six tasks.

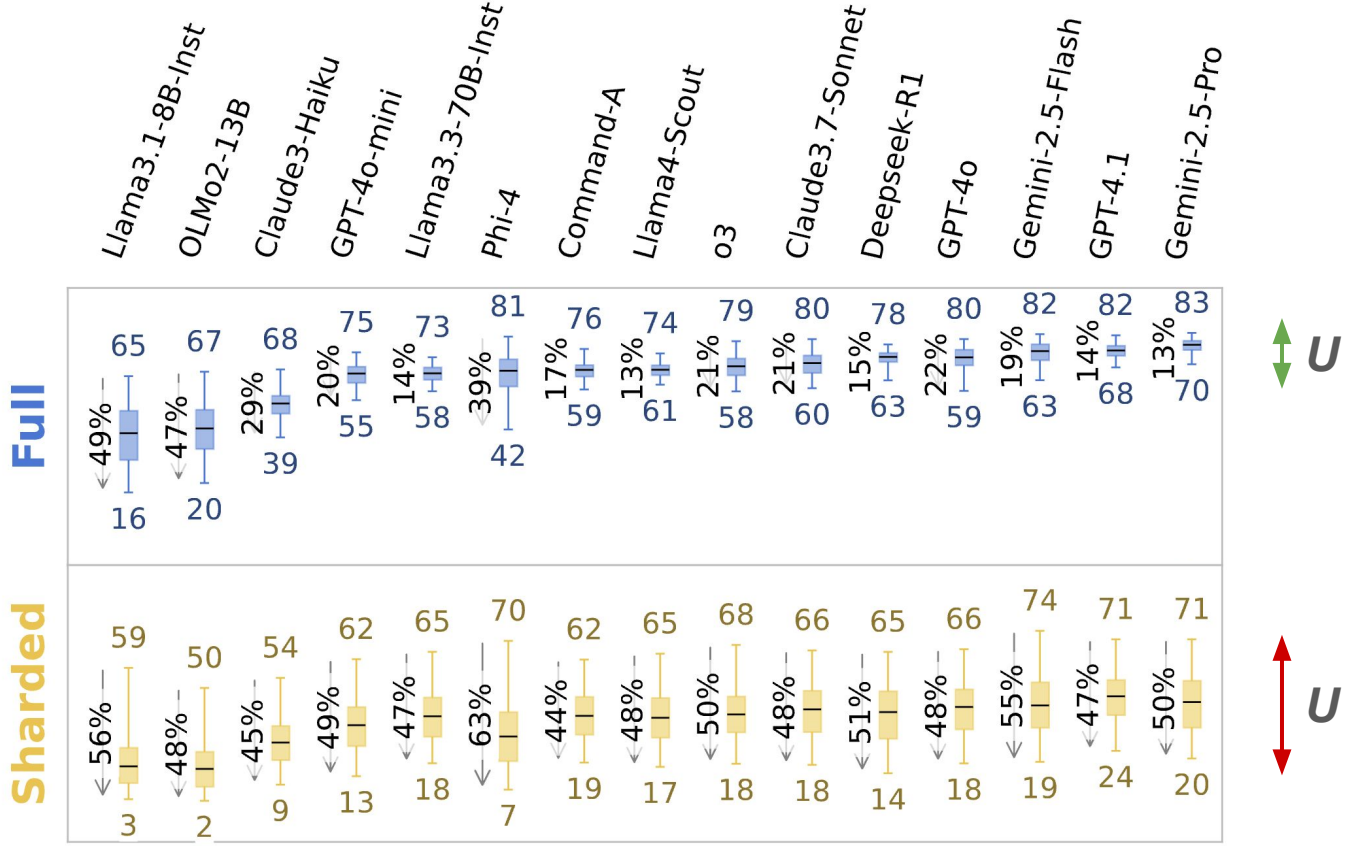
Digging deeper into scores



最良性能の落ち幅はゆるやか



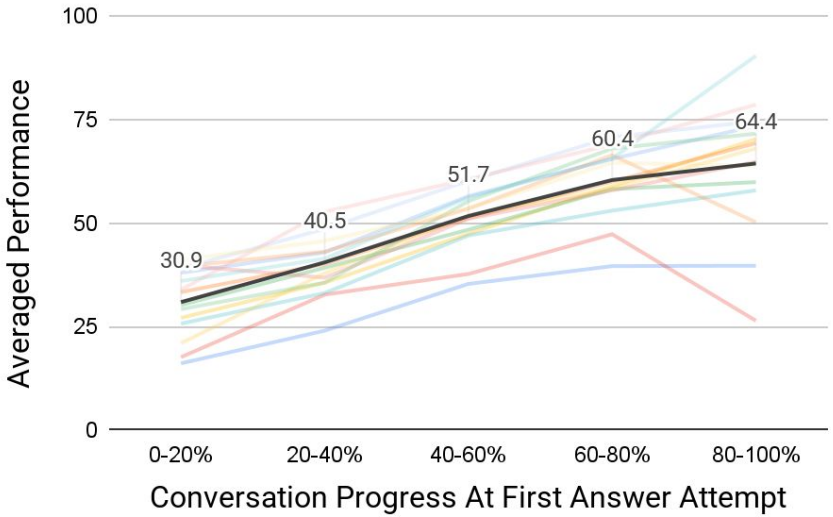
不安定性は大幅に上昇


Llama3.1-8B-Inst
OLMo2-13B
Claude3-Haiku
GPT-4o-mini
Llama3.3-70B-Inst
Phi-4
Command-A
Llama4-Scout
o3
Claude3.7-Sonnet
Deepseek-R1
GPT-4o
Gemini-2.5-Flash
GPT-4.1
Gemini-2.5-Pro
Full
49% 65 16
47% 67 20
29% 68 39
20% 75 55
14% 73 58
39% 81 42
17% 76 59
13% 74 61
21% 79 58
21% 80 60
15% 78 63
22% 80 59
19% 82 63
14% 82 68
13% 83 70
Sharded
56% 59 3
48% 50 2
45% 54 9
49% 62 13
47% 65 18
63% 70 7
44% 62 19
48% 65 17
50% 68 18
48% 66 18
51% 65 14
48% 66 18
55% 74 19
47% 71 24
50% 71 20
U
U

Analysis

LLMは指示が完全になるまで待てない

- 最初の回答が遅いほど平均性能は上がる (当たり前)
- LLMは不完全な指示を(勝手に)補完して、なるべく早く回答 してしまう

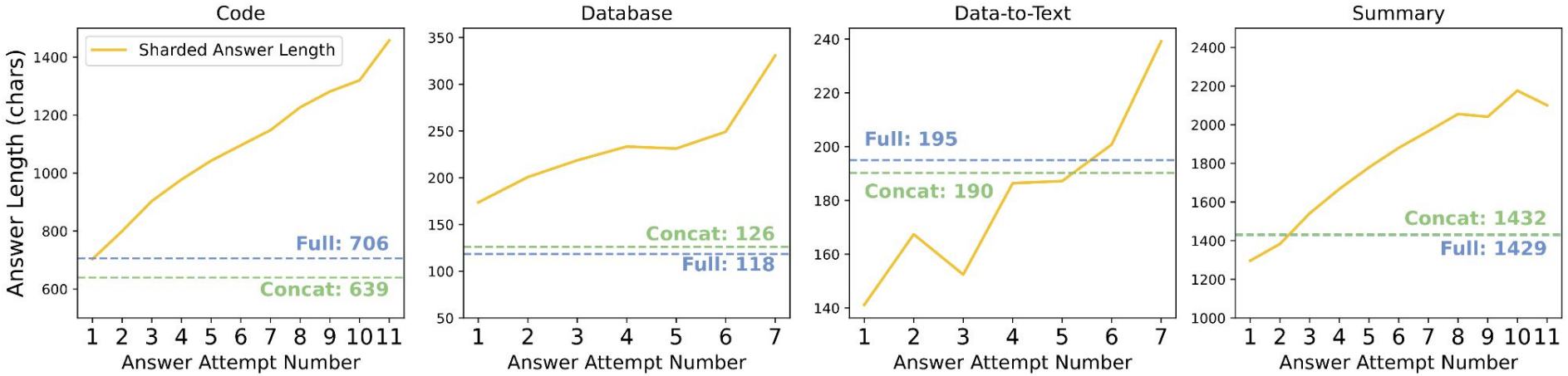


*Code & Math only

Analysis

LLMの回答は、する度に長くなっていく

- LLMは間違った回答を(なかなか)無視できず、間違いを元に新たな回答をする
  - 「いちからやり直し」がLLM本体には備わっていない



Analysis

冗長な回答ほど質が下がる

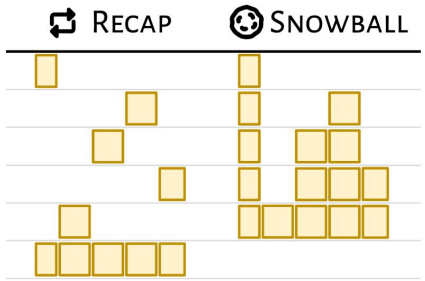
- 多くのタスクにおいて回答が短いほど性能が高い
  - 短い回答は、モデルが混乱していないことの表れ
  - 短い回答は、ユーザーにとっても読みやすい

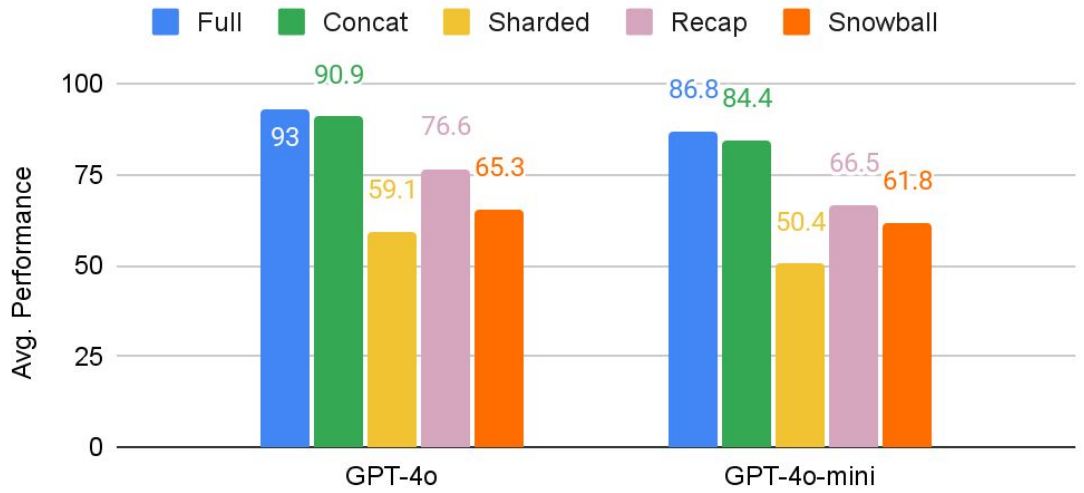
	Relative Assistant Verbosity				
Task	0-20%	20-40%	40-60%	60-80%	80-100%
Assistants responses are ...	shortest	short	median	long	longest
Code	55.3	52.3	48.9	46.9	42.5
Math	62.9	64.0	62.1	60.9	56.1
Database	43.8	40.0	37.3	34.3	31.3
Actions	41.5	49.6	54.2	53.6	50.8
Data-to-Text	25.0	24.3	24.0	23.1	21.8
Summary	15.4	14.7	13.5	12.0	10.3
Average	40.7	40.8	40.1	38.6	35.6

Implication

Agentならマルチターンの性能低下は防げるか

- 単純にshardを再投入するだけでは、大きな改善は見られない
  - Assistant が自身の過去の応答によって混乱する
- より複雑な forget / recall 戦略、Harnessで理論上は防げる





Implication

不安定性はサンプリング由来？

- Shardedでは`t = 0`であっても不安定性は改善しない
  - 応答のばらつきが会話を重ねるごとに急速に累積する
- 制御できないばらつきを考慮しても、なお大きな改善余地がある
  - e.g. Inference stack, MoE由来

シングルターン

Simulation	4o-mini AT=1.0	4o-mini AT=0.5	4o-mini AT=0.0	4o AT=1.0	4o AT=0.5	4o 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

Unreliability with various User Temp. and Assistant Temp.

Implication

LLMにshardで与えたらどんなタスクも性能は落ちるか

- タスクが分解不可能な場合に限られる
  - 新たに入ってくる情報に応じて出力を更新していく Refinementタスクでは"lost in conversation"は見られない
- 文書レベル翻訳タスクで Sharding を検証
  - 文チャンクごとに順番に翻訳しても、概ね良好な BLEU を得られる
  - Shardedによる性能劣化は見られなかった

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

Conclusion

- マルチターンの情報を適切に結合させてタスクを遂行できる LLM はまだない
- 最も大きな影響は不安定性にあり、LLMサービスにとっては大きな問題
  - e.g.,「同じ質問に対して10人中5人にはひどい応答が返る」
- LLM は過去のターンに過度に依存する傾向があり、その結果:
  - 性能が低下
  - 後続の回答が冗長化
- 短い回答は、モデルにとってもユーザーにとっても望ましい
- Harnessで外側からパッチできる(はずだ)が、LLM本体の能力としてマルチターンの対応はしっかりしてほしい