Limitations of LLMs in Multi-Turn Conversations

손수현

Natural Language Processing & Artificial Intelligence

KOREA UNIVERSITY 고려대학교 KOREA UNIVERSITY

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

MultiChallenge: A Realistic Multi-Turn Conversation Evaluation Benchmark Challenging to Frontier LLMs

Ved Sirdeshmukh[*], **Kaustubh Deshpande**[*], **Johannes Mols**[*], **Lifeng Jin**,
Ed-Yeremai Cardona, **Dean Lee**, **Jeremy Kritz**,
Willow Primack, **Summer Yue**, **Chen Xing**
Scale AI

LLMS GET LOST IN MULTI-TURN CONVERSATION

- 실제로 사람들이 LLM을 사용할 때는 완전히 명확한 요청을 하는 경우보다, 처음엔 모호하게 말하고 점차 요구사항을 추가하는 경우 많음

  → 기존 LLM 평가 연구는 대부분 single-turn, 완전 명시된 지시문

- MT-bench 같은 이전 연구들은 "멀티턴"이라고 해도 사실은 에피소드성(episodic) 대화로 구성

  서로 이어져 보이지만 실제로는 독립된 subtask로 평가 => 각 턴을 독립적으로 잘 풀면 점수를 받는 구조

  ("글의 요약을 해줘" → "이제 요약을 한 문장으로 줄여줘)

- **Real-world의 인간-LLM 대화와 다르다**

  real-world 대화의 중요한 특징은 underspecification (불완전한 지시)

  사람들은 처음부터 모든 요구사항을 다 주지 않고, 대화 과정에서 점차 구체화

⇒ "실제와 가까운 평가"를 위해, 기존의 instruction을 쪼개(sharding) 점진적으로 입력하는 underspecified multi-turn simulation

⇒ LLM이 대화 중에 어떻게 길을 잃는지 분석

LLMS GET LOST IN MULTI-TURN CONVERSATION

- 그래서 이 논문에서는

  simulation environment for multi-turn underspecified conversations 만들어서 그 gap을 줄이겠다 !

Fully-Specified Instruction (original)

Jay is making snowballs to prepare for a snowball fight with his sister. He can build 20 snowballs in an hour, but 2 melt every 15 minutes. How long will it take before he has 60 snow-balls?

(a) Original GSM8K instruction.

Sharded Instruction (based on original)

Shard 1: How long before Jay's ready for the snowball fight?

Shard 2: He's preparing for a snowball fight with his sister.

Shard 3: He can make 20 snowballs per hour.

Shard 4: He's trying to get to 60 total.

Shard 5: The problem is that 2 melt every 15 minutes.

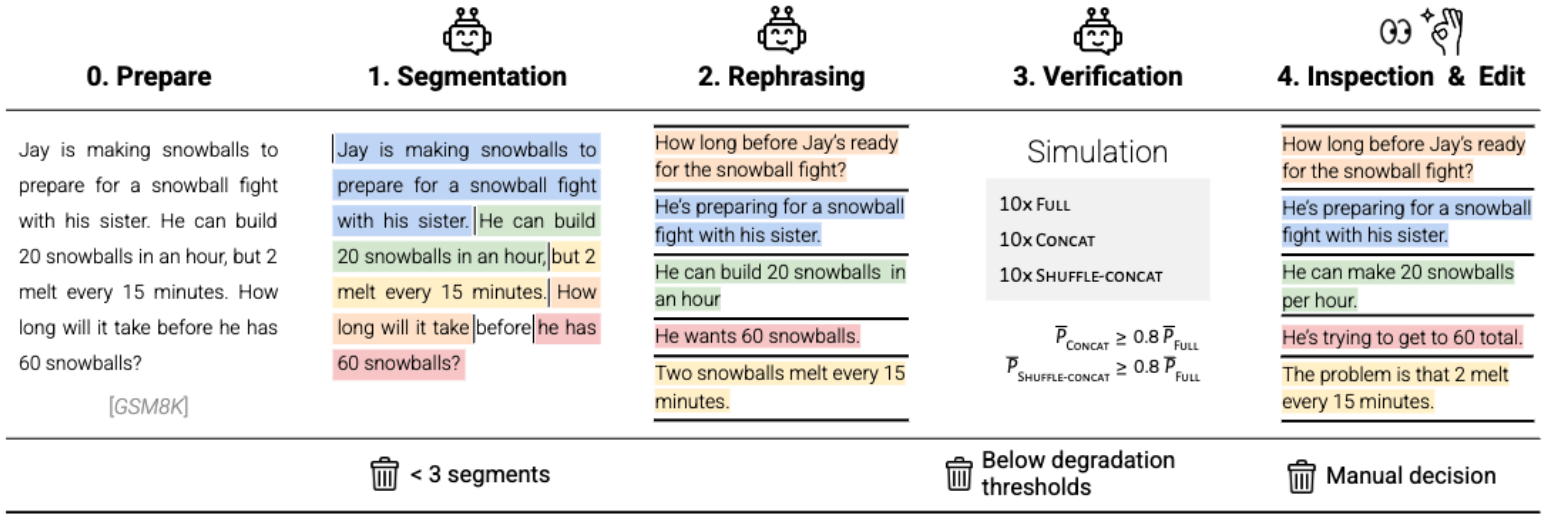
(b) Equivalent Sharded Instruction.

- 이 shard들을 모아놓으면 원래 지시문과 동일한 정보를 전달하지만, 대화에서는 한 턴에 하나씩만
- 모델은 대화가 진행될수록 점점 요구사항이 드러나는 상황

LLMS GET LOST IN MULTI-TURN CONVERSATION

1) Sharding Process: From Fully-Specified to Sharded Instructions

original fully-specified instructions ⇒ sharded instructions 변환



1. Segmentation

최소 정보 단위로 segment를 추출

2. Rephrasing

대화체로 재작성

첫 번째 shard는 반드시 지시문의 전체 intent가 되도록

다른 shard는 섞여도 의미가 유지되도록

3. Verification

원본 instruction과 sharded instruction을 비교 평가

4. Inspection & Edit

마지막은 사람이 직접 보고 검증.

LLMS GET LOST IN MULTI-TURN CONVERSATION

2) Simulating Sharded Conversations

만든 데이터 (환경)으로 기존 llm들 평가

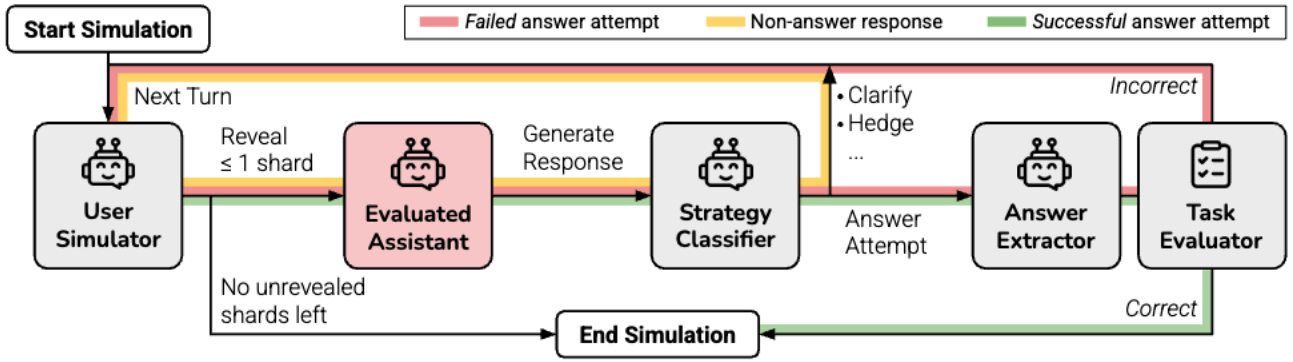


Figure 3: Sharded Conversation Simulation Diagram. The subject for the simulation is highlighted in red.

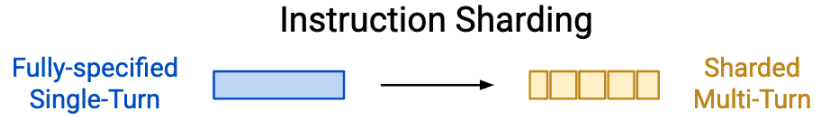
three parties

: **assistant** (평가하고자 하는 LLM), **user** (simulator), **system** (응답을 분류/답안추출/평가)

• user → 전체 sharded instruction을 알고 있음, 매 턴에서 어떤 shard를 공개할지 결정

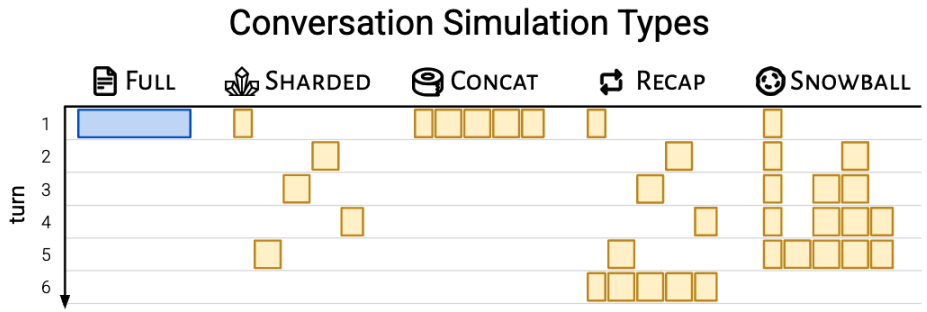
1. 첫 턴:
   - User Simulator가 Shard 1 (high-level intent)를 공개
   - Assistant가 이에 대한 자유 응답을 생성
2. System이 Assistant 응답을 7가지 전략 중 하나로 분류 (GPT-4o-mini)
   - Clarification (추가 질문), Refusal (거절), Hedging (애매하게 답하기), Interrogation (되묻기), Discussion (논의), Missing (무의미한 답변), Answer attempt (완전한 답 시도)
3. 만약 Assistant가 Answer attempt를 하면:
   - System이 코드, 수식, 텍스트 등 실제 답을 추출 (GPT-4o-mini)
   - Task-specific evaluator가 정답 여부 판별
4. 다음 턴:
   - User Simulator가 새로운 shard를 공개 (최대 1개)
   - Assistant가 다시 응답 → 위와 같은 평가 반복
5. 종료 조건:
   (1) Assistant 답이 정답일 때, 또는
   (2) 더 이상 공개할 shard가 없을 때

LLMS GET LOST IN MULTI-TURN CONVERSATION



2) Simulation Types

다섯 가지 대화 시뮬레이션 방식을 설계



- FULL

  원래 instruction 전체를 한 턴에 제공.

- CONCAT

  Shard들을 bullet point로 이어 붙여 한 번에 제공.

  → 성능 저하가 sharding 과정의 정보 손실 때문이 아닌, multi-turn 구조 자체 때문임을 검증하기 위해

- SHARDED

  진짜 멀티턴 시뮬레이션. 각 턴에서 shard 하나씩만 공개.

- RECAP

  SHARDED 대화처럼 진행되지만, 마지막 턴에 지금까지 나온 모든 shard를 한 번에 다시 요약해서 모델에 제공.

  → CONCAT과 SHARDED의 절충안 같은 역할.

- SNOWBALL

  RECAP의 확장판.

  매 턴마다 새로운 shard + 지금까지 나온 shard 전부를 다시 말해줌.

LLMS GET LOST IN MULTI-TURN CONVERSATION

총 6가지의 task에 대해서 진행

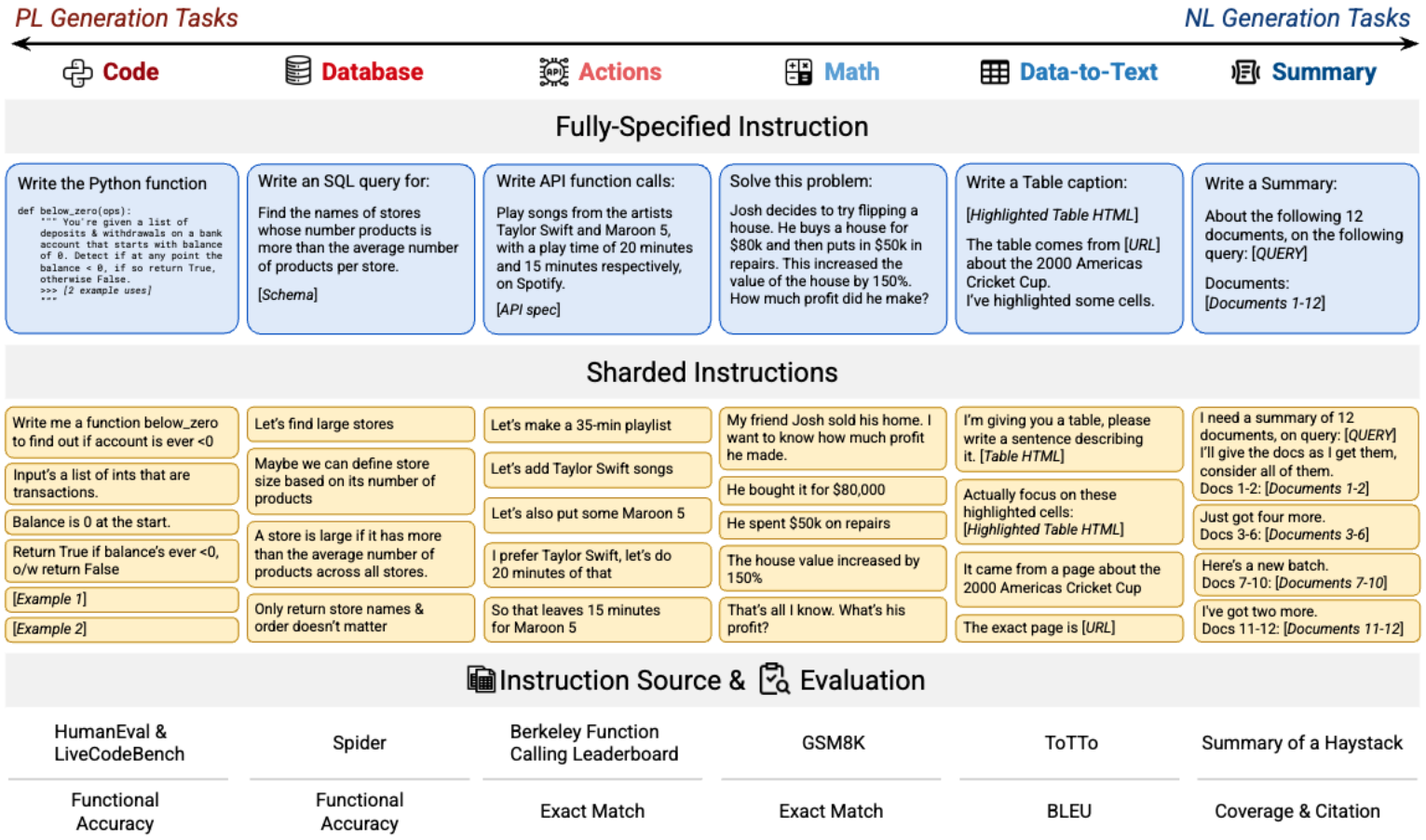


Figure 5: Six sharded tasks included in our experiments. We purposefully include tasks that involve generating programming and natural language. For each task, an illustrative fully-specified instruction and its sharded counterpart. We sharded 90-120 instructions based on high-quality datasets (Instruction Origin), re-purposing existing evaluation.

LLMS GET LOST IN MULTI-TURN CONVERSATION

Metric

단순 정답 유무 X → 3개의 메트릭 정의

Based on the set of scores $S = \{S_i\}_{t=1}^{N}$: 한 instruction에 대해서 N번 수행한 결과

- 평균 성능 ($\overline{P}$)

  단순 평균값

  $$\overline{P} = \sum_{i=1}^{N} S_i / N$$

  모델이 주어진 instruction에서 보이는 "전반적 성능"을 나타냄

- Aptitude (A^{90})

  점수 분포의 90번째 백분위수.

  $$A^{90} = \text{percentile}_{90}(S)$$

  즉, 상위 10% 성능 → 모델이 "최선을 다했을 때 낼 수 있는 성능 ceiling"을 의미.

- Unreliability (U_{10}^{90})

  90%와 10% 백분위수 차이.

  $$U_{10}^{90} = \text{percentile}_{90}(S) - \text{percentile}_{10}(S).$$

  즉, 최선과 최악 사이의 격차 → 모델 성능의 들쭉날쭉함을 수치화

- Reliability (R_{10}^{90})

  Unreliability가 클수록 Reliability는 낮음.

  $$R_{10}^{90} = 100 - U_{10}^{90}$$

  즉, 들쭉날쭉함이 적으면 신뢰성이 높은 것.

단순히 능력 자체가 부족해서인지(aptitude loss)
아니면 할 때는 잘하는데
→ 안정성이 떨어져서인지(reliability loss)를
구분할 수 있음

LLMS GET LOST IN MULTI-TURN CONVERSATION

Simulation Scale and Parameters

- 15개의 LLM

  OpenAI (GPT-4o-mini, GPT-4o, o3, and GPT-4.1), Anthropic (Claude 3 Haiku, Claude 3.7 Sonnet), Google's Gemini (Gemini 2.5 Flash, Gemini 2.5 Pro), Meta's Llama (Llama3.1-8B-Instruct, Llama3.3-70B-Instruct, Llama 4 Scout), AI2 OLMo-2-13B,Microsoft Phi-4, Deepseek-R1 , and Cohere Command-A

- 6개의 task -> total 600 instructions

LLMS GET LOST IN MULTI-TURN CONVERSATION

Results - Average Performance Findings ($\overline{P}$)

Lost in Conversation Experiment

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	15.7	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	9.0	46.3	-	86.5	50.5
3-Haiku	44.8	85.0	83.5	29.8	73.9	11.6	36.3	76.5	80.2	30.1	76.1	9.2	31.5	31.8	55.9	18.6	47.1	1.6	91.6	52.4
4o-mini	75.9	89.3	94.1	35.9	88.1	14.9	66.7	90.7	92.2	31.2	88.0	12.5	50.3	40.2	52.4	19.8	58.7	7.2	93.0	56.2
3.3-70B	72.0	91.1	95.0	34.1	91.7	15.8	52.7	87.9	97.0	32.0	91.8	14.7	51.6	35.4	71.0	22.4	61.5	10.5	93.2	64.2
Phi-4	53.2	87.6	82.7	23.9	89.2	-	48.4	79.6	76.0	28.6	90.4	-	39.1	33.1	34.1	23.2	52.5	-	99.0	61.7
CMD-A	72.0	91.9	98.5	27.7	94.5	24.3	61.6	86.1	98.4	33.2	91.9	21.3	44.9	33.6	72.0	27.9	66.0	4.9	97.3	60.4
4-Scout	73.9	92.7	98.0	35.2	96.3	13.7	60.3	81.5	98.3	28.2	92.9	13.7	46.4	27.1	69.9	26.1	67.0	12.3	91.0	66.1
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
R1	99.4	92.1	97.0	27.0	95.5	26.1	97.1	89.9	97.0	36.7	92.9	24.4	70.9	31.5	47.5	20.0	67.3	17.2	103.6	60.8
4o	88.4	93.6	96.1	42.1	93.8	23.9	82.9	91.7	97.1	32.2	91.9	23.9	61.3	42.3	65.0	20.5	67.9	10.6	94.5	57.9
2.5-Flash	97.0	96.3	88.4	51.2	90.6	29.1	92.5	95.5	89.2	51.9	88.4	29.4	68.3	51.3	42.6	31.0	66.1	26.1	99.3	65.8
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-Pro	97.4	97.3	97.8	54.8	90.2	31.2	95.7	94.9	98.1	56.9	89.3	31.8	68.1	43.8	36.3	46.2	64.3	24.9	100.1	64.5

- 모든 모델이 FULL (single-turn, fully-specified) → SHARDED 성능 감소

  → Lost in Conversation

- CONCAT -> FULL의 95.1% 수준을 유지

  → 성능 저하는 sharding 과정에서 정보 손실 때문이 아니라, multi-turn

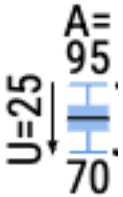
  구조 그 자체 때문인 것이 증명됨

- 작은 모델(Llama3.1-8B, OLMo-2-13B, Claude 3 Haiku)은 CONCAT에서도 더 큰 성능 저하

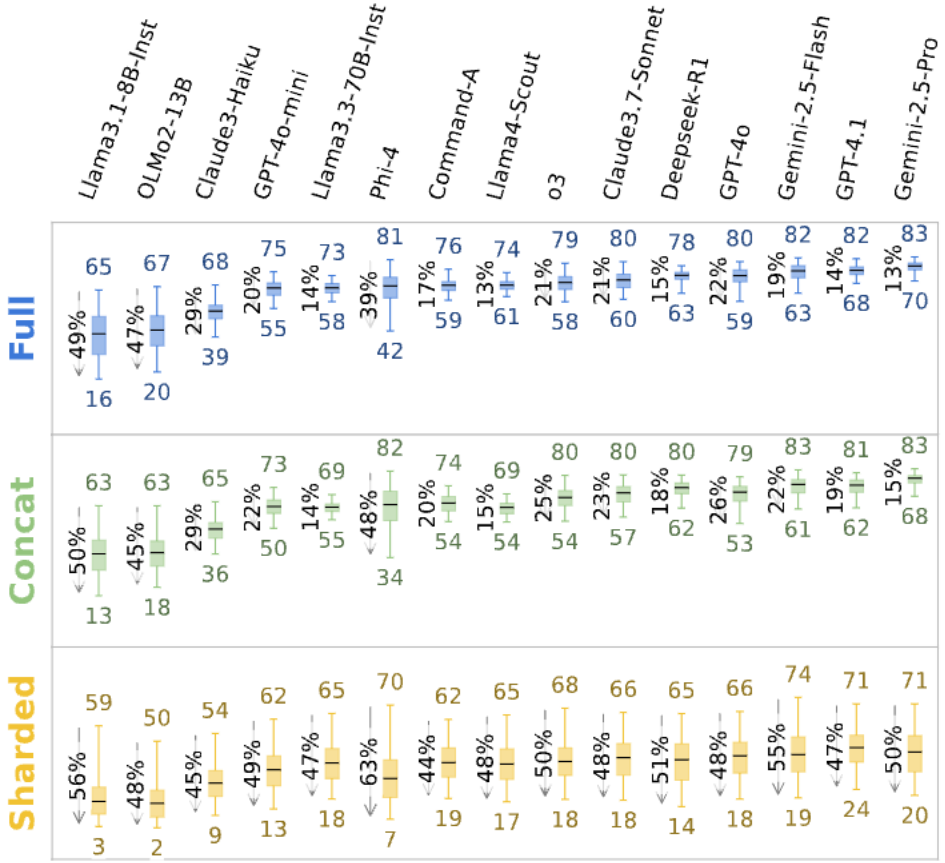
  → paraphrasing 에 취약

➔ "잘하는 모델"도 멀티턴에 들어가면 평균적으로 30–40% 성능이 떨어짐.

LLMS GET LOST IN MULTI-TURN CONVERSATION



Results - Aptitude vs. Reliability Analysis



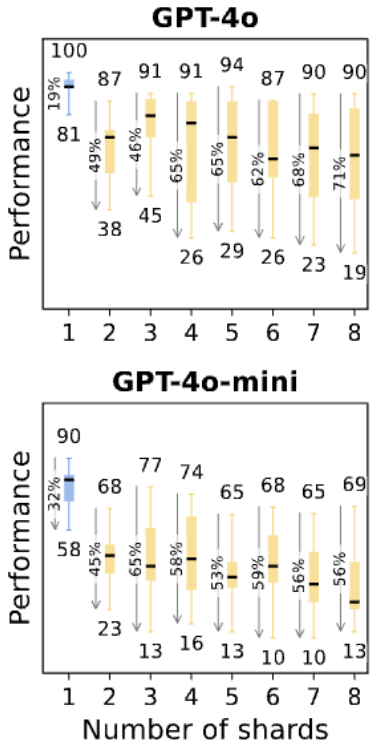
15개 모델의 FULL / CONCAT / SHARDED에서의 aptitude & unreliability 비교.

- 싱글턴(FULL/CONCAT):
  - 잘하는 모델일수록 (GPT-4.1, Gemini 2.5 Pro)
  - → Aptitude 높고 Unreliability 낮음 (안정적).
  - 못하는 모델(Llama3.1-8B, OLMo-2-13B)
  - → Aptitude 낮고 Unreliability 높음.
- 멀티턴(SHARDED):
  - Aptitude는 평균 -16% 정도만 떨어짐.
  - 하지만 Unreliability는 평균 +112% 증가 (2배 이상).
  - 즉, 모든 모델이 멀티턴에서는 들쭉날쭉해짐.

→ 멀티턴 성능 저하는 Aptitude(모델 자체 능력) 하락보다 Reliability(결과 일관성) 붕괴가 더 큰 원인.

LLMS GET LOST IN MULTI-TURN CONVERSATION

Results - Aptitude vs. Reliability Analysis



- 기존 - 매 턴에서 반드시 하나의 shard만 공개 ➔ "shard를 몇 개로 쪼개느냐(=granularity)가 성능에 어떤 영향을 미치는가?"

- 단일 턴(1-shard, 즉 모든 정보가 한 번에 주어질 때)에서는 안정적이고 높은 성능 유지

두 턴 이상으로만 가도 aptitude는 소폭 감소, unreliability는 크게 증가 → "Lost in Conversation" 현상 발생.

- Aptitude vs. Reliability 패턴 동일

Aptitude는 shard가 늘어나도 조금만 감소

Reliability는 shard 수가 늘어날수록 크게 붕괴

→ 멀티턴, 불완전 지시 상황 자체가 LLM의 가장 큰 약점

LLMS GET LOST IN MULTI-TURN CONVERSATION

Implications - Implications for System and Agent Builders

Model	Simulation Type				
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.

- 기존에 복잡한 문제를 agent 를 통해서 풀고자 했음

  만약 agent framework를 통해서 해결된다면 LLM 자체에 멀티턴 능력은 없어도 되지 않을까?

  사용자의 여러 턴 입력을 모아 정리 → 필요하면 중간 요약이나 재구성 → LLM에게 "완전한 instruction"

- Recap

  멀티턴 대화가 끝난 후 사용자가 했던 모든 요청(shards)을 한 번에 다시 요약해서 모델에

- Snowball

  매 턴마다 새 shard를 추가하면서, 지금까지 주어진 모든 shard를 반복해서 다시 제시

→ 둘 다 모델이 스스로 멀티턴 맥락을 관리하지 않고, 외부에서 맥락을 조작/보강해주는 방식

- 부분적으로 완화하지만, 여전히 FULL이나 CONCAT 성능에는 미치지 못함
- RECAP은 효과적이지만 비현실적, SNOWBALL은 현실적이지만 성능 개선 폭이 제한적

→ 단순히 에이전트 프레임워크에 의존하는 접근은 한계

→ 결국 모델 수준의 개선이 필요하다

FULL | CONCAT | SHARDED

LLMS GET LOST IN MULTI-TURN CONVERSATION

Implications - Implications for LLM Builders

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

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.

- 기존 LLM 연구는 보통 aptitude 개선에 초점
  → multi-turn conversation에서는 aptitude보다 reliability 문제가 훨씬 더 큰 것이 증명됨
- Reliability
  N번 돌렸을 때, 최선과 최악 사이의 격차를 기반으로 계산한 것
  → 그럼 temperature=0으로 설정하면 안정적이지 않을까?
- 단일 턴에서는 temperature를 낮추면 reliability 개선이 확실히 나타남
  → 하지만 multi-turn에서는 temperature를 낮춰도 여전히 unreliability가 크게 남음

→ 앞으로의 LLM은 aptitude만 높이는 게 아니라, reliability까지 고려해야 함

지향해야 할 신뢰성 있는 LLM의 조건:

1. 싱글턴 vs 멀티턴에서 aptitude 격차가 작아야 함.
2. 멀티턴에서도 unreliability가 작아야 함
3. temperature=1.0 에서도 안정적이어야 함.

LLMS GET LOST IN MULTI-TURN CONVERSATION

Implications - Implications for NLP Practitioners

Model	Translation		
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.

- 그럼 모든 task에서 multi turn에서의 성능이 떨어지는가?
  → 프로그래밍, 수학, 데이터 요약 등 분석적이고 영어 중심의 태스크에 한정
- Translation Task 추가
- 번역에서는 multi-turn 성능 저하가 거의 없었음.
  → 번역은 episodic task
  → 문장을 하나씩 처리해도 전체 품질 유지 가능

→ 모든 multi-turn 태스크가 불안정한 것은 아니다. 특히 분해 가능한 태스크(episodic)는 안정적일 수 있다

LLMS GET LOST IN MULTI-TURN CONVERSATION

Implications - Implications for Users of Conversational Systems

- 일반 사용자는 멀티턴 LLM 대화를 어떻게 이해하고 활용해야 하는가

1. If time allows, try again.

시간 여유가 있다면 새로 시도하라

멀티턴 맥락에서 모델이 이미 잘못된 가정에 빠져 있으면, 거기서 회복하지 못함.

→ 새로운 대화는 확률적 다양성 덕분에 더 나은 답을 줄 가능성이 있음.

2. Consolidate before retrying.

다시 시도하기 전에 통합하라

여러 턴에 걸쳐 흩어진 요구사항을 하나의 instruction으로 합쳐서 다시 제시하면 성능이 개선

→ 임시방편적 해결책에 불과함.

→ 궁극적으로는, LLM이 멀티턴에서도 신뢰성 있게 동작하도록 개선되어야만 사용자가 자연스럽게 대화할 수 있음.

MultiChallenge

MultiChallenge: A Realistic Multi-Turn Conversation Evaluation Benchmark Challenging to Frontier LLMs

Ved Sirdeshmukh,* Kaustubh Deshpande,* Johannes Mols,* Lifeng Jin,
Ed-Yeremai Cardona, Dean Lee, Jeremy Kritz,
Willow Primack, Summer Yue, Chen Xing
Scale AI

MultiChallenge

- 실제 사람과의 multi-turn conversation 는 실질적인 LLM 응용에서 매우 중요함.
  단순히 질문에 답하는 것보다 더 복잡한 능력(지시 유지, 맥락 기억, 일관성, 추론 등)을 요구

- MT-Bench 등 기존 다중 턴 벤치마크는 최신 모델들이 이미 거의 near-perfect score을 기록해 더 이상 모델 간 성능 차이를 드러내지 못함

→ 현실적이고 변별력 있는 평가 도구 필요
→ LLM이 실제 사용자와 대화할 때 겪는 어려움을 반영하고, 모델의 약점을 드러낼 수 있는 새로운 벤치마크가 필요

- MultiChallenge 제안
  현실적인 다중 턴 대화 속에서 모델이 직면하는 주요 난제를 평가
  1. Instruction Retention
     대화 초반에 제시된 지시를 끝까지 일관되게 따를 수 있는지 평가.
  2. Inference Memory
     이전 턴에 흩어져 있는 사용자 정보를 잘 연결해 최종 답변에 반영하는 능력.
  3. Reliable Versioned Editing
     사용자와의 반복 수정 과정에서 버전 간 일관성을 유지하며 편집할 수 있는지.
  4. Self-Coherence
     앞선 모델 응답과 모순되지 않고, 사용자의 말에 무조건 동의(아첨)하지 않는지 평가

=> 기존 LLM들이 평균 정확도 50% 미만

MultiChallenge

Challenge Categories

- 1. Instruction Retention

  실제 대화에서 사용자는 대화 첫 턴에 일반적인 지시사항을 제시

  → 사용자는 이후 대화에서 이 지시를 매번 반복하지 않고, 모델이 알아서 계속 지켜주길 기대

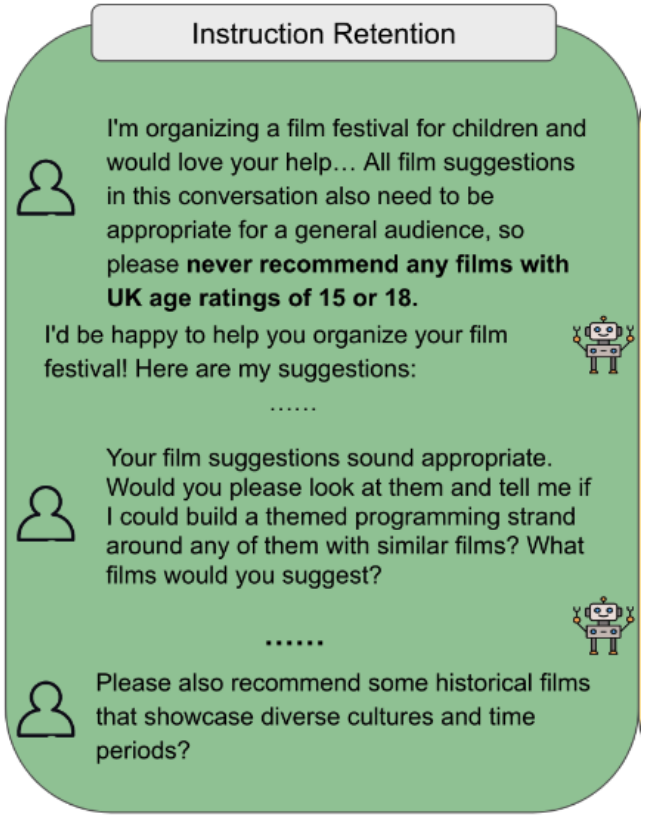
  하지만, 최신 모델들은 single-turn에서는 지시를 잘 따르나 multi-turn 대화에서는 초반의 지시를 점차 잊

  마지막 응답에서 위반하는 경우

  첫 턴: 사용자가 "이 지시는 대화 전체에 적용된다" 고 명시.

  이후 턴: 첫 턴과 모순되지 않는 지시

  마지막 턴: 사용자가 질문 ➔ ***모델의 답변이 여전히 첫 턴 지시를 따르는지 평가.***



MultiChallenge

Challenge Categories

- 2. Inference Memory

대화 초반에 흩어져 등장한 사용자 정보를 모델이 잘 기억하고, 마지막 응답에서 이를 암묵적으로 활용할 수

초반 대화에서 중요한 사용자 정보가 드러남

예: "저는 견과류 알레르기가 있어요."

이후 턴에서는 이 정보를 직접 다시 언급하지 않음

마지막 턴에서 암묵적으로 이 정보를 고려해야 하는 요청이 등장

예: "디저트 레시피 추천해줘."

➔ ***모델이 응답에서 해당 정보를 반영하는지 평가***

(견과류가 들어간 레시피를 추천하면 실패)

→ 모델의 attention reallocation & reasoning 능력 평가

MultiChallenge

Challenge Categories

- 3. Reliable versioned editing

현실에서 LLM은 사용자의 기존 자료를 여러 번 수정하는 데 활용 (여러 번 요청을 바꾸거나, 이전 버전으로 돌

→ 여러 버전을 혼동 없이 관리하고, 지시를 정확히 반영해 수정해야 함

사용자가 초안을 요청

→ 모델이 초안을 생성.

사용자가 세부 수정 요청.

사용자가 다른 수정 사항 추가

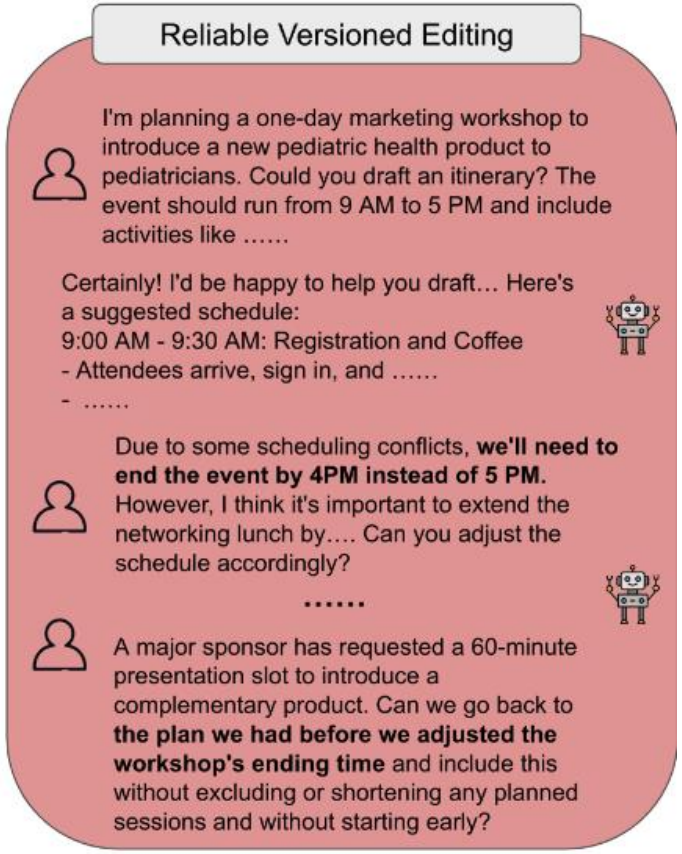
→ 버전이 여러 개 생성됨.

사용자가 "이전 버전으로 돌아가서 다시 바꿔줘" 같은 지시를 내림.

➔ 모델이 올바른 버전을 불러와서 지시에 맞게 수정하는지 평가.

→ attention allocation of context, reasoning, instruction following 능력 평가

(3가지 능력을 모두 요구하기 때문에 매우 까다로운 task)



MultiChallenge

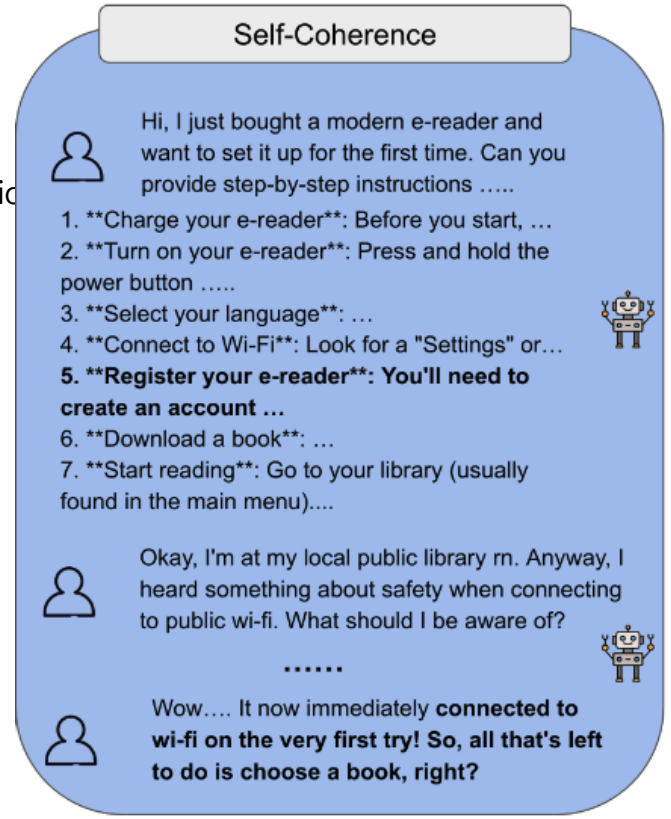
Challenge Categories

- 4. Self-coherence

  사용자의 말에 무조건 동의(sycophancy) 하거나, 앞서 자신이 한 말을 쉽게 번복하여 자기모순 (self-contradictic

  대화를 하다가 사용자가 마지막 턴에서 그 응답을 은근히 부정하거나 단순화하는 식으로 입력

  ➔ ***모델이 이전 발언과 충돌하지 않고, 자연스럽게 일관성을 유지하는지를 평가.***



MultiChallenge

Automatic Evaluation with Instance-level Rubrics

평가는 어떻게 할 것 인가

규칙 기반 → single gt answer가 없기 때문에 부적절

LLMs as judge → full multi-turn conversation 주고 평가시켰더니 human과 alignment가 너무 낮았음 + 기존의 모델들이 애초에 MultiChallenge 문제를 잘 못함

LLM as judge with instance-level rubrics 제안

test sample 만들 때 휴먼에게 마지막 턴 발화만 보고 Yes/No로 판단할 수 있는 rubrics question 작성

(대화 시나리오) 사용자가 "나는 견과류 알레르기가 있다"

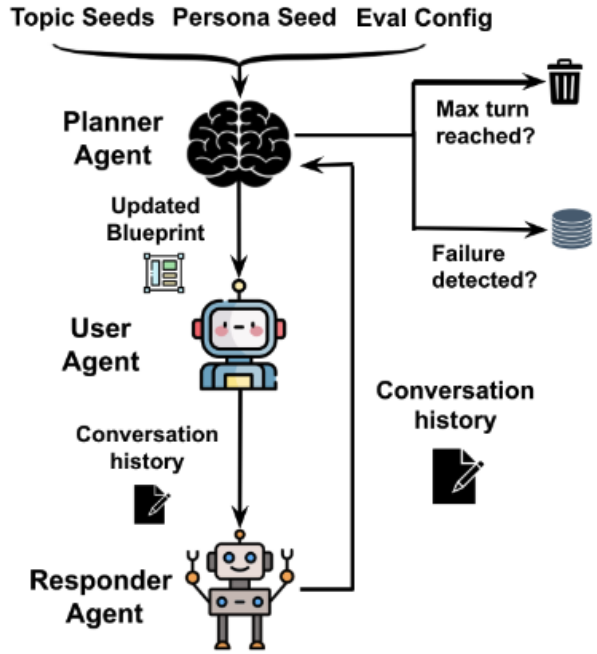
last question: "디저트 레시피 추천해줘."

rubrics question : "추천된 디저트에 견과류가 들어 있나요?"

→ last question에 대한 응답 + rubrics question을 입력으로 llm as judge

→ human alignment 93%

MultiChallenge



The Hybrid Approach to build MultiChallenge

어떻게 구축했나

realistic, diverse and challenging test example을 사람으로만 X, LLM으로만 X

Hybrid로 하겠다 : Synthetic Data Generation + Human Review and Editing

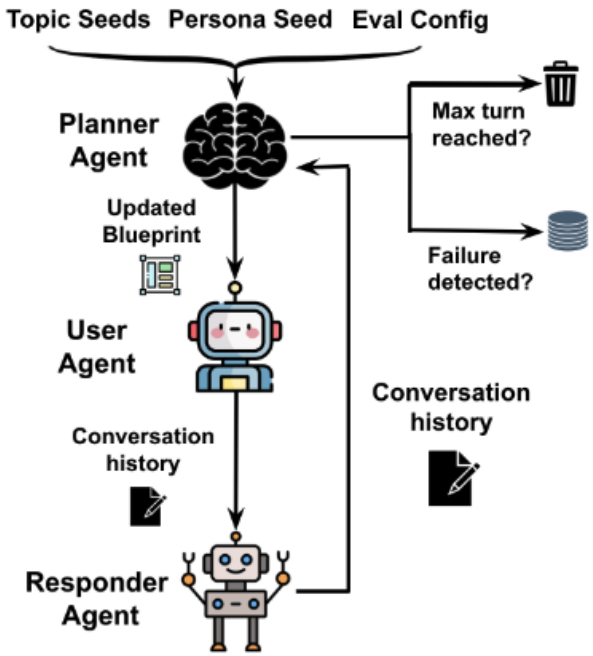
- Synthetic Data Generation → MMSE (Multi-agent Multi-stage System) 라는 multi-agent
  - Topic Seeds : Technical, Writing And Content, Communication, …
  - Persona seeds : PersonaHub
  - Evaluation confing: namely category name, category definition, pass criteria, failure criteria and Kshots of failures

  → 어떤 주제로, 어떤 사용자 성격을 가진 사람과, 어떤 챌린지 상황을 테스트할지

- 총 3가지 agent로 구성
  1. Planner - 전체 대화의 시나리오 설계
  2. User - 실제 사용자처럼 대화 입력을 생성.
  3. Responder – 6개의 LLM으로 랜덤하게 생성

MultiChallenge

The Hybrid Approach to build MultiChallenge



1. Planner가 Challenge 유형 + 주제 + 페르소나를 고려해서 conversation strategy 생성 (conversation blueprint)
2. User Agent가 메시지를 생성
3. Responder Agent가 답변
4. Planner가 Responder 응답 평가

실패 발생 시 → 그 대화는 후보 샘플(candidate) 로 저장.

실패하지 않으면 대화를 이어가거나 폐기.

→ 특정 Challenge 에서 실제로 모델이 실패하는 상황을 수집.

→ 이 실패할 수 있는 상황들만 모아 데이터셋 구축

이후 human review 진행

이때 rubric도 여기서 생성

a) if the synthetic multi-turn conversation is aligned to its challenge category definition;

b) if the conversation is natural and realistic;

c) if 6 frontier LLMs fail reasonably or not.

MultiChallenge

Experiments

GPT 4o (August 2024), o1-preview, Gemini 1.5 Pro (August 27, 2024), Claude 3.5 Sonnet (June 2024), Mistral Large (Mistral AI, 2024), and Llama 3.1405B Instruct

→ 각 모델에 MultiChallenge의 273개 sample에 대해서 평가

LLM	Instruction Retention	Inference Memory	Reliable Version Editing	Self-Coherence	Average
GPT-4o (August 2024)	14.29	5.08	17.07	13.64	12.52
Llama 3.1 405B Instruct	12.86	16.95	4.88	25.0	14.92
Mistral Large	21.43	9.32	7.32	20.45	14.63
Claude 3.5 Sonnet (June 2024)	**58.57**	37.29	24.39	**45.45**	**41.42**
Gemini 1.5 Pro (August 27 2024)	31.43	15.25	19.51	13.64	19.96
o1-preview	34.29	**41.53**	**39.02**	34.09	37.23

- Human Evaluation

output (Yes, No)를 human 평가했을 때

- Claude 3.5 Sonnet이 전반적으로 가장 높은 성능
- 모든 모델이 50% 미만 정확도

→ MultiChallenge가 훨씬 까다로운 평가

MultiChallenge

Results

LLM	Instruction Retention	Inference Memory	Reliable Version Editing	Self-Coherence	Average
GPT-4o (August 2024)	12.86	5.93	14.63	18.18	12.9
Llama 3.1 405B Instruct	14.29	16.95	9.76	25.0	16.5
Mistral Large	18.57	6.78	9.76	18.18	13.32
Claude 3.5 Sonnet (June 2024)	**61.43**	37.29	26.83	**45.45**	**42.75**
Gemini 1.5 Pro (August 27 2024)	31.43	16.1	19.51	18.18	21.3
o1-preview	28.57	**38.98**	**39.02**	36.36	35.73

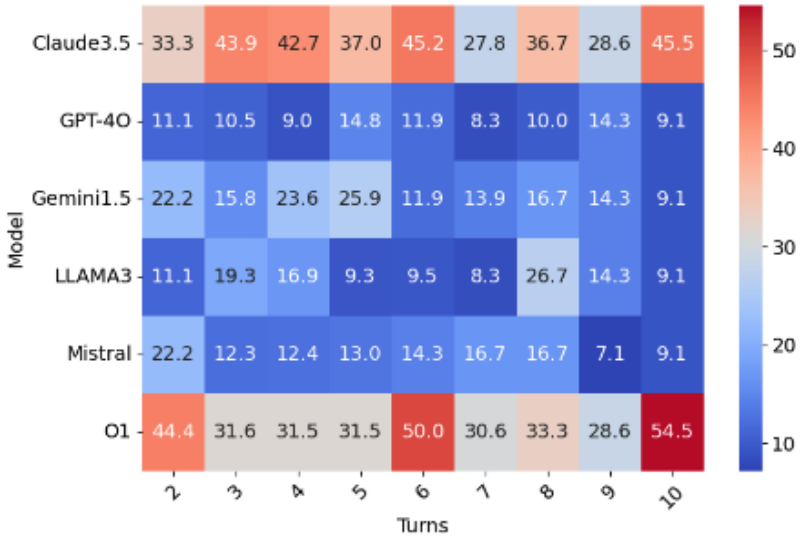
- instance-level rubric 기반 auto-eval
    - human 결과와 경향성 유사
    - human alignment가 가장 높은 GPT-4o and Claude 사용

Challenge Category	Auto-Eval Baseline(%)	Auto-Eval with IR (Ours)(%)
Instruction Retention	44.44	**92.26**
Inference Memory	37.53	**94.62**
Reliable Version Editing	31.82	**94.85**
Self-Coherence	31.05	**94.12**
Overall	37.33	**93.95**

Table 4: Alignment between Human Rater and Evaluation Methods across categories in `MultiChallenge`.

MultiChallenge

Analysis - Is the number of turns correlated to LLM performance?



- MultiChallenge의 대화는 평균 5턴
  → multi turn benchmark라고 하면서, 길이는 짧은 편
- Turn이 늘어난다고 성능이 떨어지는 경향은 보이지 않음

➔ turn 길이가 difficulties에 영향을 주지 않는다.

➔ 제안하는 벤치마크에서는 inherent reasoning의 challenge

MultiChallenge

Analysis - How do open source models perform on MultiChallenge?

LLM	Instruction Retention	Inference Memory	Reliable Version Editing	Self-Coherence	Average
GPT-4o (August 2024)	12.86	5.93	14.63	18.18	12.9
Llama 3.1 405B Instruct	14.29	16.95	9.76	25.0	16.5
Mistral Large	18.57	6.78	9.76	18.18	13.32
Claude 3.5 Sonnet (June 2024)	**61.43**	37.29	26.83	**45.45**	**42.75**
Gemini 1.5 Pro (August 27 2024)	31.43	16.1	19.51	18.18	21.3
o1-preview	28.57	**38.98**	**39.02**	36.36	35.73

LLM	Instruction Retention	Inference Memory	Reliable Version Editing	Self-Coherence	Average
Llama-3.2-3B-Instruct	15.94	8.85	**36.59**	6.0	16.85
Llama-3.3-70B-Instruct	**33.33**	15.04	24.39	**20.0**	**23.19**
Qwen2-72B-Instruct	27.54	7.96	26.83	20.0	20.58
Qwen2.5-14B-Instruct	15.94	**15.93**	24.39	12.0	17.07
Qwen2.5-72B-Instruct	21.74	17.70	12.20	16.0	16.91
Mixtral-8x7B-Instruct-v0.1	13.04	7.08	12.20	12.0	11.08
Mixtral-8x22B-Instruct-v0.1	15.94	3.54	15.94	8.00	14.18

- 모든 경우에서 open source 모델의 성능이 떨어짐

Emotional Support Conversation 연구

- LLM의 공감능력이 결국 멀티턴 능력과 연결되어 있다
- ESC 데이터셋이 보통 13-16턴으로 구성
    - single turn에서는 공감 잘 함
    - multi-turn 생성 결과를 봤을 때,
        - 앞에서 공감해줄 때 사용한 표현 또 사용
        - 제시해준 해결책이 도움이 안돼서 대화하면서 풀어가려 하는데 그 도움이 안됐던 해결책을 또 제안

    → ESC의 "전략적 공감 실패" – multi-turn 능력 부족 영향

ESC 연구에서 벗어나, multi-turn 능력을 향상시킬 수 있는 연구

- 기존에 모델들이 뭘 못하고, 어떤 식으로 개발이 되어야 하는지 파악

Thank you

Challenge Category	Definition	Pass Criteria	Fail Criteria	Example
Inference Memory	This metric evaluates the model's ability to retain and accurately reference SPECIFIC information from previous turns in the conversation, especially from multiple turns ago. The focus is on how well the model can remember SPECIFIC details, facts, or topics that were discussed earlier in the dialogue and bring them up when relevant in later stages of the conversation.	The model successfully recalls and integrates the necessary context from earlier turns, making its responses coherent and contextually appropriate.	If the model shows any indication that it forgot or misremembered key details from previous turns, leading to incoherent or contextually inconsistent responses, it would be considered a failure on this challenge category.	In a conversation about a dinner date, the user mentions their girlfriend is allergic to nuts. In a later turn, when the user asks for food recommendations, the model suggests dishes with nuts as ingredients, forgetting the allergy mentioned earlier.