TReMu: Towards Neuro-Symbolic Temporal Reasoning for LLM-Agents with Memory in Multi-Session Dialogues

Anonymous ACL submission

Abstract

Temporal reasoning in multi-session dialogues presents a significant challenge which has been under-studied in previous temporal reasoning benchmarks. To bridge this gap, we propose a new evaluation task for temporal reasoning in multi-session dialogues and introduce an approach to construct a new benchmark by augmenting dialogues from LoCoMo and creating multi-choice QAs. Furthermore, we present TReMu, a new framework aimed at enhancing the temporal reasoning capabilities of LLM-agents in this context. Specifically, the framework employs *time-aware memorization* through timeline summarization, generating retrievable memory by summarizing events in each dialogue session with their inferred dates. Additionally, we integrate *neuro-symbolic temporal reasoning*, where LLMs generate Python code to perform temporal calculations and select answers. Experimental evaluations on popular LLMs demonstrate that our benchmark is challenging, and the proposed framework significantly improves temporal reasoning performance compared to baseline methods, raising from 29.83 on GPT-4o via standard prompting to 77.67 via our approach and highlighting its effectiveness in addressing temporal reasoning in multi-session dialogues.[1]

1 Introduction

In the context of multi-session dialogues, temporal reasoning is both critical and challenging for LLM-agents. As dialogue sessions proceed, storing and retrieving relevant information efficiently becomes more difficult (Maharana et al., 2024), such as failing to retrieve specific temporal details from long history and dialogues exceed the input limit of LLMs. Additionally, research has shown that LLMs overlook important contextual information from long dialogue histories due to the accumulation of irrelevant historical data, referred to as "historical noise" (Wang et al., 2023). These challenges underscore the need for enhanced temporal reasoning capabilities in LLM-agents for effective handling of multi-session dialogues.

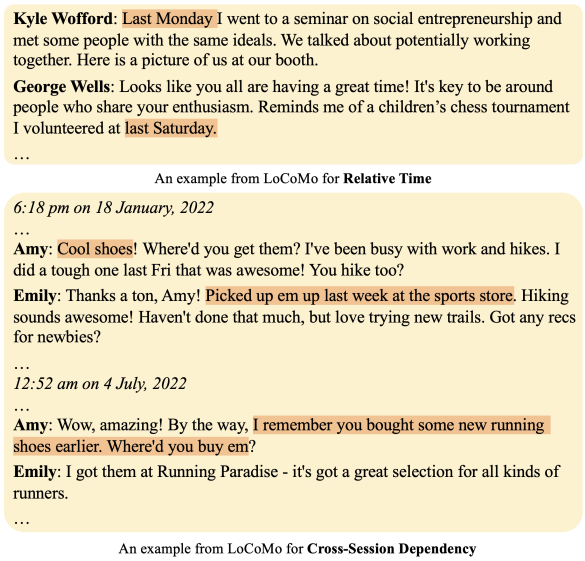

Figure 1: Examples from LoCoMo showing the two temporal characteristics we focus on in this work.

However, most existing temporal reasoning benchmarks cannot be used directly for this study, because they are usually built on shorter texts, such as stories and Wikipedia articles, that contain clear temporal information (Chen et al., 2021; Wang and Zhao, 2024; Xiong et al., 2024). Even benchmarks designed for dialogues, like TimeDial (Qin et al., 2021) and LoCoMo (Maharana et al., 2024), do not explicitly consider the special temporal characteristics in multi-session dialogues, such as *relative time* and *cross-session dependency*. For instance, speakers often use relative time expressions instead of specific dates, requiring the model to infer exact event times. Moreover, it is common for speakers to recall past events from previous sessions, creating cross-session dependencies, where events from different sessions involve the same or related entities and reflect changes over time. This further

[1]We will release our data and code upon paper acceptance for reproducibility.

requires LLMs to retain context effectively when reasoning about events across multiple sessions.

In this work, we present **TReMu** (**T**emporal **Re**asoning for LLM-Agents in **Mu**lti-Session Dialogues), a novel framework designed to enhance temporal reasoning in multi-session dialogues. Our framework introduces **time-aware memorization**, which uses timeline summarization to generate summaries for each dialogue session, identifying events and associating them with their inferred dates. These summaries, linked to specific times (either session times or inferred event dates), serve as retrievable memory. This effectively addresses events expressed in relative time by distinguishing when such an event occurred from when it was mentioned by the speaker.

During reasoning, we propose a **neuro-symbolic temporal reasoning** approach inspired by recent work that integrates LLMs with symbolic reasoning, translating questions into symbolic language before using a solver to find answers (Pan et al., 2023; Olausson et al., 2023). Specifically, given a temporal question, we retrieve relevant memory and instruct the LLMs to generate Python code. This approach leverages the LLMs' strong Python coding capabilities and existing Python libraries for temporal calculations. The generated code serves as an intermediate rationale. By executing the code line-by-line, the model follows step-by-step reasoning similar to CoT (Wei et al., 2022), leading the model to select the correct answer.

Due to the absence of temporal reasoning evaluation benchmarks specific to multi-session dialogues, we propose a method to construct a new evaluation benchmark focusing on two key temporal characteristics: *relative time* and *cross-session dependency*. By augmenting dialogues from LoCoMo (Maharana et al., 2024), we create multiple-choice temporal questions spanning three types of reasoning to evaluate the temporal reasoning capabilities of LLMs in this context.

We evaluate our framework based on three popular LLMs—GPT-4o, GPT-4o-mini, and GPT-3.5-Turbo—on our benchmark. The results show that our benchmark is challenging, revealing suboptimal performance for LLMs. In contrast, our framework demonstrates superior performance compared to baseline methods, such as CoT, highlighting the effectiveness of our approach in improving temporal reasoning in multi-session dialogues.

Our contributions are as follows:

- We propose a new framework for temporal reasoning in multi-session dialogues, integrating time-aware memorization and neuro-symbolic temporal reasoning.
- We propose a method to construct a temporal reasoning evaluation benchmark for multi-session dialogues by augmenting an existing dataset, explicitly covering the temporal characteristics of relative time and cross-session dependency.
- Through extensive experiments, we empirically show that temporal reasoning in multi-session dialogues poses significant challenges for LLMs, even with strategies like CoT. However, our framework significantly improves LLMs' temporal reasoning in this context.

2 Benchmark Construction

In this section, we introduce the construction pipeline to build our temporal QA benchmark for evaluating LLM-agents' temporal reasoning in multi-session dialogues. As mentioned earlier, we mainly focus on the two temporal characteristics in multi-session dialogues: **relative time** and **cross-session dependency**.

2.1 Benchmark Design

We propose augmenting an existing dataset to create a benchmark for evaluating LLM-agents' temporal reasoning in multi-session dialogues. After a thorough examination, we selected LoCoMo (Maharana et al., 2024), which comprises dialogues averaging 600 turns and 16,000 tokens across up to 32 sessions. In comparison to existing multi-session dialogue datasets, LoCoMo features the longest dialogues and the most sessions (as shown in Table 1), thus presenting a greater challenge.

As mentioned earlier, our benchmark focuses on two key temporal characteristics in multi-session dialogues: *relative time* and *cross-session dependency*. To achieve this, we follow previous benchmarks (Chen et al., 2021; Xiong et al., 2024; Wang and Zhao, 2024) by creating temporal QA pairs based on temporal events in the dialogues. Specifically, we design each temporal QA based on either a single event or a pair of events:

• **Single Event**: We select events expressed with relative time and develop a temporal reasoning type called *Temporal Anchoring*, which asks for the exact time of the event.

Dialogue Dataset	Avg. Turns Per Conv.	Avg. Sessions Per Conv.	Avg. Tokens Per Conv.	Time Interval	Collection
MSC (Xu et al., 2022)	53.3	4	1,225.9	few days	Crowdsourcing
Conversation Chronicles (Jang et al., 2023)	58.5	5	1,054.7	few hours - years	LLM-generated
LoCoMo (Maharana et al., 2024) (Ours)	304.9	19.3	9,209.2	few months	LLM-gen. + crowdsourcing

Table 1: Statistics of the chosen multi-session dialogue dataset, LoCoMo, compared to others.

Question Type	Count	# of Options	# of Events	Event Type
Temporal Anchoring	264	5	1	relative time
Temporal Precedence	102	3	2	cross session dependency (+ relative time)
Temporal Interval	234	5	2	cross session dependency (+ relative time)
Total	600	–	–	–
- Unanswerable	112	–	–	–
LoCoMo (Maharana et al., 2024)	321	–	–	–

Table 2: Dataset statistics and details of the constructed benchmark.*# of Options* refers to the number of options for each temporal question. *# of Events* refers to the number of selected events to create each temporal question. *Event Type* specifies the type of temporal events chosen for question creation, where *(+ relative time)* indicates that relative time was an additional consideration in event selection. Note that our benchmark not only contains more temporal QAs than LoCoMo, but also include unanswerable questions.

• **Two Events**: We choose pairs of relevant events from different sessions that exhibit cross-session dependency. We also consider relative time as an extra factor to increase the complexity of the questions. Two temporal reasoning types are applied: *Temporal Precedence*, which asks which event occurred first, and *Temporal Interval*, which asks for the duration between the two events.

2.2 Construction Pipeline

To construct our benchmark, we follow the design of our benchmark and utilize a systematic step-by-step approach with GPT-4o. The prompt for each step is shown in Appendix § A.

Step 1: Temporal Event Extraction We begin by prompting GPT-4o to extract all temporal events from each dialogue session. In addition, we instruct GPT-4o to annotate the relative time expressions for these events, facilitating the selection process for creating temporal QAs.

Step 2: Temporal Event Linking Next, we link the extracted events containing cross-session dependency within the dialogue. We prompt GPT-4o with the extracted events and instruct it to group those related to the same or relevant entities across different sessions, particularly those reflecting changes in attributes over time. For example, the event "Debra Ryan told her mentor about her business idea" from an early session is linked to "Debra Ryan started her own business" from a later session.

Step 3: Temporal QA Creation Once the temporal events are processed, we prompt GPT-4o to select those events that meet the criteria for different temporal reasoning types and generate multiple-choice temporal QAs. Additionally, we create unanswerable questions, as in prior QA benchmarks (Rajpurkar et al., 2018), to comprehensively assess models' temporal reasoning capabilities.

Step 4: Quality Control We observe various noises in generated QAs, such as incorrect inferences about exact times. To ensure the benchmark's quality, we follow recent temporal reasoning benchmarks for LLMs, such as TGQA (Xiong et al., 2024), to perform quality control. We manually review each question to verify that it aligns with our design specifications and that the answers are correctly grounded in the dialogue. We also revise well-constructed questions with incorrect answers and remove any unreasonable ones. The final temporal QA benchmark covers time intervals from days to months and its statistics and details are presented in Table 2. Particularly, our final benchmark not only contains more temporal QAs than LoCoMo, but also include unanswerable questions, which are not covered in LoCoMo. We also include examples of QAs for different temporal reasoning types in Appendix §. B.

3 Methodology

3.1 Preliminary: Memory-Augmented LLM-Agents

To address the limit of LLMs struggling in retaining information from long input text, recent studies turn to equip LLM agents with memory to support long-turn conversations (Lu et al., 2023; Packer et al., 2023; Zhong et al., 2024). Therefore, we base our study on memory-augmented LLM-agents.

The general pipeline of memory-augmented LLM-agents comprises three stages: *memorization*, *retrieval*, and *response*. In the *memorization* stage, the model summarizes each dialogue session and stores these summaries as memory. During the *retrieval* stage, the model retrieves the most relevant memory for the current dialogue session. This retrieved memory is then concatenated with the ongoing dialogue to generate the next *response*. Specifically, we build our framework based on MemoChat (Lu et al., 2023), which realizes this three-stage process through prompting and has demonstrated effectiveness in handling long-range dialogues.

3.2 TReMu

Building on the memory-augmented LLM-agent pipeline, we introduce our framework called **TReMu** as shown in Algorithm 1. The framework consists of two key components: *time-aware memorization* and *neuro-symbolic temporal reasoning*.

Algorithm 1 TReMu

Initialize:
Time-aware Memorization Model LLM_{mem}
Memory Retrieval Model $\text{LLM}_{\text{retrieval}}$
Neuro-symbolic Reasoning Model LLM_{code}
Symbolic Solver $\mathcal{P}$
Memorization pool $\mathcal{M} \leftarrow \emptyset$
{*Time-aware Memorization*}
for each dialogue session d_i in dialogue $\mathcal{D}$ **do**
  $m_i \leftarrow \text{LLM}_{\text{mem}}(d_i)$
  $\mathcal{M} \leftarrow f_{org}(\mathcal{M}, m_i)$
end for
{*Neuro-symbolic Temporal Reasoning*}
for each temporal question q **do**
  $m_{retrieved} \leftarrow \text{LLM}_{\text{retrieval}}(q, \mathcal{M})$
  $c \leftarrow \text{LLM}_{code}(q, m_{retrieved})$
  $o \leftarrow \mathcal{P}(c)$
  final answer $a \leftarrow \text{LLM}(q, o)$
end for

3.2.1 Time-aware Memorization

Our time-aware memorization builds on timeline summarization (Steen and Markert, 2019; Rajaby Faghihi et al., 2022; Sojitra et al., 2024) and it consists of two steps: *Temporal Memory Writing* and *Memory Organization*. During Temporal Memory Writing (prompt in Appendix §.C), we instruct LLM agents to generate memory pieces while also extracting and associating mentioned events with inferred dates. Unlike prior approaches that summarize entire sessions holistically, our method produces fine-grained memory pieces linked to specific inferred time markers. As shown in Tables 3 and 4, our memorization outputs memory pieces corresponding to events with inferred time steps that facilitates to mitigate temporal ambiguity. For example, the highlighted texts show that Michelle cooked a meal and later referenced cooking it at different dates. This enables finer temporal granularity, effectively distinguishing events based on inferred time intervals.

Topic	Summary
Catching Up	Daniel and Michelle catch up on new events in their lives including new jobs, hobbies, and activities.
Hobbies and Daily Rituals	Daniel and Michelle discuss their hobbies and rituals like running, ballet, playing guitar, meditation, and cooking.
Cooking and Celebrations	Both talk about their cooking experiences and celebrate Daniel's promotion.
Books and Recommendations	Michelle and Daniel discuss books they've read and recommend some to each other.
Personal Items with Sentimental Value	Michelle and Daniel talk about cameras and a vintage motorcycle with sentimental value.

Table 3: Output memory from MemoChat based on one dialogue session in LoCoMo.

Time	Summary
01/28/2020	Daniel and Michelle share updates on their lives... including Michelle starting her Masters in Psychology, Daniel starting a new job where he learns to code and problem-solve, and Michelle's hobby of ballet, meditation, and journaling... **Michelle mentions she made an Italian meal last Saturday** and Daniel made salsa for a taco night. Daniel also shared receiving a promotion ...
01/27/2020	Daniel received a promotion and celebrated with a dinner at his favorite spot.
01/25/2020	**Michelle cooked a delicious Italian meal for her friends**, including pasta, garlic bread, and tiramisu.
01/24/2020	Daniel made a huge batch of salsa and hosted a taco night with friends.
01/20/2020	Michelle started her Masters in Psychology.

Table 4: Output memory from Time-aware Memorization based on the same dialogue session in LoCoMo.

Then, we perform Memory Organization on the output memory pieces to maintain long-term memory. We structure memory in a timeline format, grouping events that occur simultaneously and indexing them based on inferred timesteps. This approach enhances the distinction between an event's occurrence and its mention, reducing temporal ambiguity and improving time-based retrieval. These enhancements mark a significant difference from traditional memorization approaches, supporting efficiency in temporal reasoning.

3.2.2 Neuro-symbolic Temporal Reasoning

Inspired by the recent progress in neuro-symbolic reasoning for LLMs (Han et al., 2022; Pan et al., 2023), we propose leveraging LLMs to translate temporal reasoning questions into Python code as intermediate rationales, which is executed as the reasoning process to derive answers (prompts shown in Appendix §.D). We tried different symbolic languages and finally chose Python because SOTA LLMs are better at generating Python code and there exist Python libraries that support temporal calculations, like *datetime* and *dateutil*. Particularly, *dateutil* provides a function *relativedelta* supporting relative time calculation, for example we can infer next Friday using *TODAY + relativedelta(weekday=FR)*. Meanwhile, we provide implemented functions to be directly called, such as "weekRange(t)" returns the start date and the end date of the week that t lies in. Different from other works in temporal reasoning based on code execution(Li et al., 2023), we enable our LLM agents with **function-calling** abilities, ensuring correctness and expanding the range of temporal reasoning tasks beyond simple precedence relations.

We provide demonstration via in-context learning to generate Python code with function calling, given then question and retrieved memory. After the generated code is executed, the output and code serve as **intermediate rationales**, and the LLM is prompted again to give the answer. Particularly, our reasoning approach offers an alternative form of CoT. While the original CoT (Wei et al., 2022) performs step-by-step reasoning in natural language, our neuro-symbolic approach conducts temporal reasoning by executing generated code line-by-line in a programming language. This neuro-symbolic method retains the core concept of CoT's step-by-step reasoning while leveraging the precision and additional support provided by Python code and packages. However, prior works (Li et al., 2023) rely solely on solver outputs without providing intermediate justifications.

4 Experiments

4.1 Experimental Setup

Models. We build our framework using various *black-box* LLMs: GPT-4o[2], GPT-4o-mini[3], and GPT-3.5-Turbo[4]. Particularly, for GPT-3.5-Turbo, many of LoCoMo dialogues are longer than its input length, we then follow LoCoMo (Maharana et al., 2024) which earlier dialogues are omitted.

Particularly, we have also tried different *open-source* LLMs but most of them cannot handle the long dialogue inputs from LoCoMo, for example only about 10% of dialogues can be fed into Llama-3-70B. And even for those shorter dialogues that can be fed into Llama-3-70B, we notice that the model gets lost and fails to follow instructions, even failing to generate in the desired format. Therefore, we leave the adaptation to LLMs as future work.

Baselines. Since in our setting of multi-session dialogues where the conversations exceed the input limits of LLMs, we consider the memory mechanism as a critical component of baselines in order to feed complete dialogue information. Therefore, we include the following baselines for comparison:

- **Standard Prompting (SP)**: The entire dialogue is provided along with each temporal question, with additional instructions for selecting the correct answer.
- **Chain-of-Thought (CoT)** (Wei et al., 2022): Similar to SP, but with additional instructions for LLMs to solve questions step-by-step.
- **MemoChat** (Lu et al., 2023): Given that multi-turn dialogues can exceed the model's input length, and since our approach builds on memory-augmented LLM-agents, MemoChat serves as a baseline where we modify the response stage to answer temporal questions.

To better understand the effectiveness of each component in our framework, we evaluate the following variants as baselines for the ablation study:

- **MemoChat + CoT**: This baseline applies CoT in the response stage to answer temporal questions step-by-step using the retrieved memory.
- **Timeline + CoT**: Based on the framework of memory-augmented LLM-agents, we modify the original memorization with our proposed timeline summarization and combine it with CoT as a baseline.

Comparing MemoChat + CoT and Timeline + CoT allows us to assess the impact of replacing the standard memory mechanism in LLM agents with our time-aware memorization. Additionally, comparing Timeline + CoT with TReMu highlights the

[2]Specifically, *gpt-4o-2024-05-13*.
[3]Specifically, *gpt-4o-mini-2024-07-18*.
[4]Specifically, *gpt-3.5-turbo-0125*.

Methods	Accuracy				Unanswerable Questions		
	TA	TP	TI	Overall	Precision	Recall	F1
SP	18.18	58.82	30.34	29.83	46.88	13.39	20.84
CoT	67.80	74.51	49.15	61.67	42.61	43.75	43.18
MemoChat	35.23	43.14	25.21	32.67	24.30	77.68	37.02
Memochat + CoT	51.14	49.02	26.50	41.67	24.80	**81.25**	38.00
Timeline + CoT	83.33	78.41	58.55	71.50	48.51	58.04	52.84
TReMu	**84.47**	**81.37**	**68.38**	**77.67**	**55.48**	76.79	**64.42**

Table 5: Experimental results of various methods based on GPT-4o. We use TA to represent Temporal Anchoring, TP for Temporal Precedence and TI for Temporal Interval.

Methods	Accuracy				Unanswerable Questions		
	TA	TP	TI	Overall	Precision	Recall	F1
SP	20.08	50.00	29.91	29.00	**40.00**	26.79	32.08
CoT	46.59	**62.75**	37.18	45.67	33.96	48.21	39.86
MemoChat	21.21	39.22	23.50	25.17	21.11	74.11	32.88
Memochat + CoT	24.62	45.10	24.36	28.00	21.11	75.00	32.94
Timeline + CoT	55.68	59.80	38.46	49.67	30.73	59.82	40.60
TReMu	**64.02**	46.08	**38.89**	**51.17**	29.21	**92.86**	**44.44**

Table 6: Experimental results of various methods based on GPT-4o-mini.

effect of replacing CoT with our neuro-symbolic reasoning approach.

Evaluation Metrics. We primarily use **accuracy** to assess the overall performance of temporal reasoning. In addition, for unanswerable questions, we calculate **precision**, **recall**, and the **F1** score to specifically measure performance on this subset of questions. Specifically, precision is computed as the accuracy of questions the model predicts as "unanswerable," while recall is determined by the accuracy of questions where the ground truth answer is "unanswerable."

4.2 Experimental Results

The results are shown in Tables 5, 6, and 7 for GPT-4o, GPT-4o-mini, and GPT-3.5-Turbo, respectively. On the recent TRAM benchmark (Wang and Zhao, 2024), existing LLMs demonstrate strong performance with direct prompting. For instance, GPT-4 achieves an accuracy of 82 using CoT, while GPT-3.5 attains 71.40. In contrast, our benchmark is significantly more challenging. GPT-4o achieves only 61.67 with CoT and 29.83 with SP, whereas GPT-3.5 performs even worse, scoring 25.83 with CoT and 23.83 with SP. These performance gaps likely stem from the complexity of multi-session dialogues and their temporal dependencies, which are not explicitly addressed in previous benchmarks.

Our framework outperforms all baseline methods across all three LLMs in terms of both accuracy and F1 scores, with a notable increase in accuracy from 29.83 with SP to 77.67 with our framework using GPT-4o. This demonstrates the effectiveness of our approach in enhancing temporal reasoning for multi-session dialogues. However, incorporating a memory mechanism performs worse than CoT for GPT-4o and GPT-4o-mini. This may be because these models have sufficient input lengths to process LoCoMo dialogues, enabling them to identify relevant temporal information without additional memory augmentation. In contrast, for GPT-3.5, which has a shorter input limit, the memory mechanism generally improves performance by allowing the model to retrieve information from memory rather than truncated dialogues.

Furthermore, we find that incorporating CoT generally improves performance, aligning with previous findings (Wang and Zhao, 2024; Xiong et al., 2024). In particular, CoT encourages models to search for relevant information within dialogues. However, due to the dialogues' length, the models sometimes generate responses like "I cannot find the mention of ...," which hinders their temporal reasoning capabilities. This further underscores the necessity of a memory mechanism to support long dialogue settings.

4.3 Ablation Study

From Tables 5, 6, and 7, the comparison between MemoChat + CoT and Timeline + CoT highlights the importance of memory representation. Time-aware memorization improves accuracy by instructing models to infer temporal information—particularly relative time—during memorization and mitigate temporal ambiguity. Furthermore, replacing CoT with symbolic reasoning, as seen in the comparison between Timeline + CoT and TReMu, leads to additional performance gains.

Methods	Accuracy				Unanswerable Questions		
	TA	TP	TI	Overall	Precision	Recall	F1
SP	21.59	31.37	23.08	23.83	22.91	46.43	30.68
CoT	23.86	38.24	22.65	25.83	20.97	50.00	29.56
MemoChat	17.42	45.10	23.50	24.50	21.93	66.96	33.04
Memochat + CoT	20.45	**53.92**	**26.50**	28.50	21.79	50.00	30.36
Timeline + CoT	32.58	44.12	22.65	30.67	22.57	51.79	31.44
TReMu	**42.42**	37.25	22.22	**33.67**	**23.33**	**75**	**35.60**

Table 7: Experimental results of various methods based on GPT-3.5-Turbo.

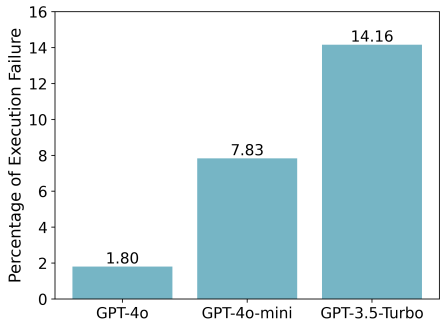

Figure 2: The percentage of execution failures.

This improvement stems from the models' ability to generate Python code while retaining the benefits of step-by-step reasoning. This aligns with recent research integrating LLMs with symbolic reasoners for various reasoning tasks (Olausson et al., 2023; Pan et al., 2023).

4.4 Execution Failure Study

We also measure the percentage of generated code that fails to execute and, during inference, we regenerate the code when such errors occur. The results, shown in Figure 2, indicate that the percentages of execution failure are generally low across all three LLMs, demonstrating the reliability of our Python-based symbolic reasoning approach. As expected, GPT-4o exhibits the lowest rate of execution failure, while GPT-3.5-Turbo has the highest, corresponding to the overall performance differences we demonstrate above in temporal reasoning among these models. This likely reflects the inherent performance gap between the LLMs.

4.5 Case Study

In this section, we demonstrate how the two key components of our framework—*time-aware memorization* and *neuro-symbolic temporal reasoning*—work in real cases. We compare the outputs of CoT and our framework based on GPT-4o.

In Figure 3, with CoT, even though GPT-4o successfully identifies that the key temporal information is "Sharon's survival course started on 12 March 2020," but it gets confused with "week-long course" and infers the end date of the course, incorrectly selecting "Unanswerable." In contrast, with our framework's time-aware memorization, the model retrieves the event from memory along with its properly inferred time. During the reasoning stage, the model utilizes this memory to distinguish between when the speaker, Sharon, mentioned the event (03/16/2020) and when the event occurred (03/12/2020). Then the model defines the corresponding variable in the generated code, i.e., *t_start_course*, to precisely capture the time.

Figure 4 illustrates another mistake made via CoT. The model correctly infers that the "last week" corresponds to the session time of 16 March 2020 but fails to match the week range with the correct answer—the week of 03/09/2020 is the week of 03/11/2020 but the model does not realize this. As for our framework, the model leverages the Python *dateutil* package's *relativedelta* function, alongside our custom *weekRange* function, to accurately infer the last week's range. This neuro-symbolic reasoning not only facilitates the model to reason step-by-step but also enhances it by incorporating external temporal functions to support more accurate temporal reasoning.

5 Related Work

Temporal Reasoning for LLMs. Recent advancements in large language models (LLMs) have brought significant improvements in reasoning capabilities (Huang and Chang, 2023), leading to growing interest in temporal reasoning (Chu et al., 2024; Qiu et al., 2024). Existing approaches primarily address this challenge through time-aware language modeling. For example, Kanashiro Pereira; Tan et al. propose fine-tuning strategies to enhance temporal reasoning, while Zhou et al.; Yang et al. introduce auxiliary objectives to incorporate external temporal knowledge. However, studies such as Chu et al.; Qiu et al. show that state-of-the-art LLMs still exhibit suboptimal performance in tem-

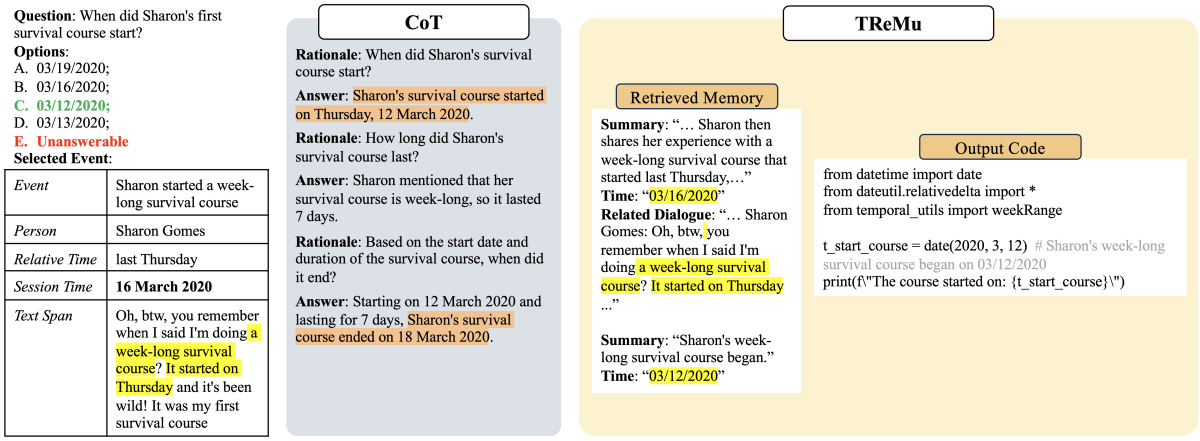

Figure 3: The first case study comparing CoT and our proposed framework, where CoT results in the wrong answer "E" but our approach selects the correct option "C". We highlight the key information in colors.

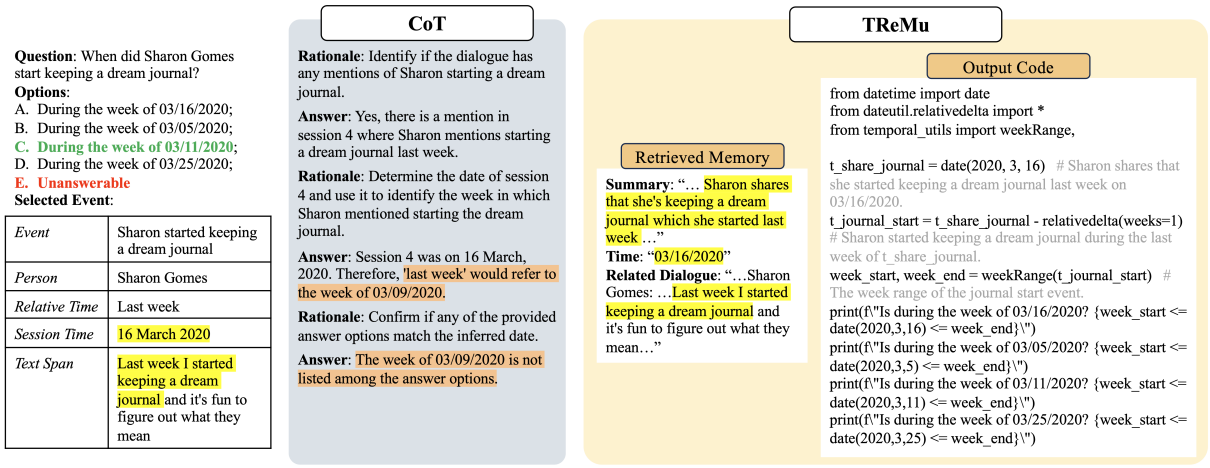

Figure 4: The second case study comparing CoT and our proposed framework, where CoT results in the wrong answer "E" but our approach selects the correct option "C". We highlight the key information in colors.

poral reasoning with prompting techniques. Our framework differs from these works by utilizing memory-augmented LLM agents, enhancing memorization through timeline summarization, and integrating neuro-symbolic reasoning as an intermediate step for answering temporal questions.

Multi-session Dialogues. Several studies have developed multi-session dialogues benchmarks. Xu et al. introduced MSC, the first multi-session dataset incorporating time intervals between sessions. Similarly, Bae et al. proposed a dynamic memory management method to maintain up-to-date user information and introduced a Korean multi-session dialogue dataset. Jang et al. created the CONVERSATION CHRONICLES dataset, designed for long-term conversations that integrate time intervals and detailed speaker relationships. More recently, Maharana et al. introduced LoCoMo, a dataset featuring long-term and multi-modal dialogues. While our work is situated within this context, it specifically targets temporal reasoning, addressing the temporal characteristics of relative time and cross-session dependency, which have not been explicitly explored in prior research.

6 Conclusion

In this paper, we address the critical challenge of temporal reasoning in multi-session dialogues for LLM-agents. We present a method to construct a temporal reasoning evaluation benchmark by augmenting dialogues from LoCoMo and covering the temporal characteristics of relative time and cross-session dependency. Furthermore, we introduce a novel framework which combines time-aware memorization through timeline summarization with neuro-symbolic temporal reasoning by translating temporal questions into executable Python code. Through extensive evaluations, we demonstrate that our benchmark presents significant challenges, and our framework substantially outperforms baseline methods in improving temporal reasoning for multi-session dialogues.

7 Limitations

Our work has several limitations:

Assessment in QA settings. Our evaluation follows the standard practice of recent temporal reasoning benchmarks like (Zhou et al., 2019; Wang and Zhao, 2024), using a multiple-choice format for reliable assessment. A more ideal setting would be evaluating in generative dialogue settings. However, generative QA evaluations pose significant challenges due to variations in temporal expressions (e.g., "January 1st" vs. "01/01"), making exact match and F1 token score unreliable. Thus, we prioritize benchmark reliability and accuracy over a more ambitious generative QA setting. We leave the extension of the current benchmark to generative dialogue settings as future work

Open-sourced Models. Our current experiments mainly focus on close-sourced models. However, as we have pointed out in our Experiment section § 4.1, we found most open-source LLMs cannot handle the long dialogue inputs from LoCoMo, for example only about 10% of dialogues can be fed into Llama-3-70B. And even for those shorter dialogues that can be fed into Llama-3-70B, we notice that the model gets lost and fails to follow instructions, even failing to generate in the desired format. We therefore consider the adaptation of our framework to open-sourced models as future work. Note that we will release our code and dataset for reproducibility.

References

Sanghwan Bae, Donghyun Kwak, Soyoung Kang, Min Young Lee, Sungdong Kim, Yuin Jeong, Hyeri Kim, Sang-Woo Lee, Woomyoung Park, and Nako Sung. 2022. Keep me updated! memory management in long-term conversations. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Wenhu Chen, Xinyi Wang, and William Yang Wang. 2021. A dataset for answering time-sensitive questions. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Zheng Chu, Jingchang Chen, Qianglong Chen, Weijiang Yu, Haotian Wang, Ming Liu, and Bing Qin. 2024. TimeBench: A comprehensive evaluation of temporal reasoning abilities in large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Bangkok, Thailand. Association for Computational Linguistics.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Luke Benson, Lucy Sun, Ekaterina Zubova, Yujie Qiao, Matthew Burtell, David Peng, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Shafiq Joty, Alexander R. Fabbri, Wojciech Kryscinski, Xi Victoria Lin, Caiming Xiong, and Dragomir Radev. 2022. Folio: Natural language reasoning with first-order logic. *arXiv preprint arXiv:2209.00840*.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. In *61st Annual Meeting of the Association for Computational Linguistics, ACL 2023*, pages 1049–1065. Association for Computational Linguistics (ACL).

Jihyoung Jang, Minseong Boo, and Hyounghun Kim. 2023. Conversation chronicles: Towards diverse temporal and relational dynamics in multi-session conversations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13584–13606.

Lis Kanashiro Pereira. 2022. Attention-focused adversarial training for robust temporal reasoning. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, Marseille, France. European Language Resources Association.

Xingxuan Li, Liying Cheng, Qingyu Tan, Hwee Tou Ng, Shafiq Joty, and Lidong Bing. 2023. Unlocking temporal question answering for large language models using code execution. *arXiv e-prints*, pages arXiv–2305.

Junru Lu, Siyu An, Mingbao Lin, Gabriele Pergola, Yulan He, Di Yin, Xing Sun, and Yunsheng Wu. 2023. Memochat: Tuning llms to use memos for consistent long-range open-domain conversation. *arXiv preprint arXiv:2308.08239*.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. Evaluating very long-term conversational memory of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Bangkok, Thailand. Association for Computational Linguistics.

Theo Olausson, Alex Gu, Ben Lipkin, Cedegao Zhang, Armando Solar-Lezama, Joshua Tenenbaum, and Roger Levy. 2023. Linc: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5153–5176.

Charles Packer, Vivian Fang, Shishir G Patil, Kevin Lin, Sarah Wooders, and Joseph E Gonzalez. 2023. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. 2023. Logic-lm: Empowering large language models with symbolic solvers for faithful

logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3806–3824.

Lianhui Qin, Aditya Gupta, Shyam Upadhyay, Luheng He, Yejin Choi, and Manaal Faruqui. 2021. Timedial: Temporal commonsense reasoning in dialog. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7066–7076.

Yifu Qiu, Zheng Zhao, Yftah Ziser, Anna Korhonen, Edoardo Ponti, and Shay B Cohen. 2024. Are large language model temporally grounded? In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7057–7076.

Hossein Rajaby Faghihi, Bashar Alhafni, Ke Zhang, Shihao Ran, Joel Tetreault, and Alejandro Jaimes. 2022. Crisisltlsum: A benchmark for local crisis event timeline extraction and summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, Online and Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789.

Daivik Sojitra, Raghav Jain, Sriparna Saha, Adam Jatowt, and Manish Gupta. 2024. Timeline summarization in the era of llms. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2657–2661.

Julius Steen and Katja Markert. 2019. Abstractive timeline summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, Hong Kong, China. Association for Computational Linguistics.

Qingyu Tan, Hwee Tou Ng, and Lidong Bing. 2023. Towards benchmarking and improving the temporal reasoning capability of large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14820–14835.

Bing Wang, Xinnian Liang, Jian Yang, Hui Huang, Shuangzhi Wu, Peihao Wu, Lu Lu, Zejun Ma, and Zhoujun Li. 2023. Enhancing large language model with self-controlled memory framework. *arXiv preprint arXiv:2304.13343*.

Yuqing Wang and Yun Zhao. 2024. TRAM: Benchmarking temporal reasoning for large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Siheng Xiong, Ali Payani, Ramana Kompella, and Faramarz Fekri. 2024. Large language models can learn temporal reasoning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Bangkok, Thailand. Association for Computational Linguistics.

Jing Xu, Arthur Szlam, and Jason Weston. 2022. Beyond goldfish memory: Long-term open-domain conversation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Dublin, Ireland. Association for Computational Linguistics.

Sen Yang, Xin Li, Lidong Bing, and Wai Lam. 2023. Once upon a $time$ in $graph$: Relative-time pretraining for complex temporal reasoning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Singapore. Association for Computational Linguistics.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19724–19731.

Ben Zhou, Daniel Khashabi, Qiang Ning, and Dan Roth. 2019. "going on a vacation" takes longer than "going for a walk": A study of temporal commonsense understanding. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3363–3369.

Ben Zhou, Kyle Richardson, Qiang Ning, Tushar Khot, Ashish Sabharwal, and Dan Roth. 2021. Temporal reasoning on implicit events from distant supervision. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1361–1371.

A Prompts for Benchmark Construction

We use GPT-4o to construct a temporal reasoning benchmark for multi-session dialogues. The first step is the temporal event extraction using the prompt shown in Figure 5. Then the prompt for the second step, temporal event linking, is shown in Figure 6. With the grouped temporal events, we use the prompt in Figure 7 to create temporal QAs.

Let's extract events based on the dialogue between two speakers.
- The extracted events are represented in the form of a JSON list without comments.
- Each entry corresponds to an extracted event, and is a dictionary containing the following keys: "event", "person", "date", "relative time", "id", "text span".
- The "event" field contains a short description of the event, and the description should be in sentences following the pattern: subject, verb, object. An example can be: "John visited a cat café ".
- The "person" field contains the speaker who did the event.
- The "relative time" field contains the mentioned relative time expression in the corresponding original text span, such as "Last week" and "two years ago". If there is no relative time mentioned, leave it as empty.
- The "date" field contains a date of the event in a format as "month/day/year". You should infer the exact date based on the mentioned relative time. If the date can't be inferred, leave it as empty.
- The "id" field contains a unique numerical identifier for the event.
- The "text span" field contains the original text span in the given dialogue corresponding to the extracted event.

Figure 5: Prompt for Temporal Event Extraction.

Your goal is to find the relevant temporal events which correspond to cross-session dependencies across the multi-session dialogues from given temporal events, and your identified relevant temporal events should reflect the situation where events in one session influence or are contingent upon events in other sessions.

Instruction:
1. Read the extracted temporal events and the whole dialogues. The extracted events for each session are represented in the form of a JSON list, and each entry corresponds to an extracted event containing the following keys: "event", "person", "date", "relative time", "id", "text span". The "event" field contains a description of the event, the "person" field contains the speaker who did the event, the "date" field contains a date of the event, the "relative time" field contains the mentioned relative time expression in the corresponding original text span, the "id" field contains a unique numerical identifier for the event, and the "text span" field contains the original text span in the given dialogue corresponding to the extracted event.
2. Identify all possible relevant temporal events corresponding to cross-session dependencies as many as you can. You should focus on temporal events that contain same or relevant entities and reflect the attribute change of the entities. For example, a decision in one session can affect a timeline in another session without clear linkage.
3. Output the all grouped temporal events in a JSON object and give each grouped events with a unique key name like "events_0". Keep the same format for each temporal event, and include the session id and session time for each temporal event, i.e, adding "session id" and "session time" fields to the entry for each temporal event. Do not include your explanation.

Figure 6: Prompt for Temporal Event Linking.

You are tasked with creating all possible challenging and complex questions with answers based on the given grouped temporal events. The given temporal events have been grouped according to cross-session dependencies, where each group is a list of relevant temporal events referred to attribute change of same or relevant entities across sessions.

Your designed questions should require temporal reasoning to answer, meaning that the answers need understanding the sequence of events across different sessions and the time elapsed between them. Do not produce questions that are answerable with commonsense only and try to make as many as you can. Meanwhile, you should design the questions can be answered in just a few words.

Step-by-step Instructions:
1. Consider Temporal Reasoning Types: Consider three temporal reasoning types for making questions: "Temporal Anchoring", "Temporal Interval" and "Temporal Precedence". Specifically, "Temporal Anchoring" involves identifying a specific point in time when an event occurred, "Temporal Interval" involves inferring the duration of between two events and "Temporal Precedence" involves determining which event occurred before another event.
2. Select Temporal Events: For each question, only select one or two relevant temporal events corresponding to the same person from the given grouped temporal events to design QAs for testing temporal reasoning. Do not make up or extract new temporal events beyond the given grouped temporal events. To ensure the complexity of questions, select events with relative time expression for "Temporal Anchoring", like "last Sunday" specified in the field "relative time", and select two relevant events from one grouped event for "Temporal Interval" or "Temporal Precedence".
3. Create Temporal Reasoning QAs: Create QAs based on your determined temporal reasoning type and selected temporal events. Do not make QAs that are answered based on sessions, like "how many sessions did it take...". Your answers should use accurate exact time instead of relative time. Also, you need to make both answerable and unanswerable questions, and answerable QAs must be factually correct while unanswerable questions should be relevant but can't be answered according to the dialogue.
4. Output QAs in a JSON list: Each entry should be a dictionary containing the following keys: "question", "answer", "selected events", "reasoning type" and "explanation".
   - The "reasoning type" field contains the reasoning type of the QA you are going to make, and it must be one of the three types, "Temporal Anchoring", "Temporal Interval" and "Temporal Precedence".
   - The "selected events" field contains a list of chosen temporal events for designing QAs, and you need to output their original metadata of events with their corresponding session id from the provided grouped temporal events.
   - The "question" and "answer" fields contain the QA designed for temporal reasoning. For unanswerable questions, leave "answer" as "unanswerable".
   - The "explanation" field contains the explanation of why the answer is factually correct.

Output the QAs in a JSON list without any other text, and here is an example. Given the three relevant temporal events "John got injured", "John joined a yoga class to help with injury", and "John felt better with the injury after doing yoga", one potential question would be "How long did it take for John to recover from the injury?"

Figure 7: Prompt for Temporal QA Creation.

B Examples of Temporal QAs in Constructed Benchmark

We show examples of final temporal QAs for different temporal reasoning types in Figure 8, 9 and 10. In each example, we highlight the ground truth answer as green and show the corresponding selected temporal events for constructing the question below the question.

C Prompt for Time-aware Memorization

We show the designed prompt for time-aware memorization in Figure 11.

D Prompt for Neuro-symbolic Temporal Reasoning

We show the designed prompts for neuro-symbolic temporal reasoning here. Specifically, we first perform memory retrieval with the prompt in Figure 12. Then we prompt to generate Python code via in-context learning as in Figure 13. With the generated code and its execution result, we finally prompt the LLM to select the answer, and the prompt is shown in Figure 14.

Question: When did Molly print out her camping trip photos?
Options: **A. 04/27/2022**; B. 05/06/2022; C. 05/15/2022; D. 04/25/2022; E. Unanswerable

Event	Molly printed out her camping trip photos
Person	Molly
Relative Time	last Wednesday
Session ID	session_6
Session Time	05/06/2022
Text Span	Last Wednesday, I printed out my camping trip photos and put them up in my art studio.

Figure 8: An example temporal QA for Temporal Anchoring

Question: Which event happened first, Grace teaching her first dance class or performing the lead role of her hip hop piece?
Options: A. Grace performing the lead role of her hip hop piece; **B. Grace teaching her first dance class**; C. Unanswerable

Event	Grace taught her first dance class
Person	Grace
Relative Time	last month
Session ID	session_6
Session Time	05/22/2022
Text Span	Had my first dance teaching class last month, it was awesome!

Event	Grace performed the lead role of her hip hop piece
Person	Grace
Relative Time	last night
Session ID	session_15
Session Time	08/26/2022
Text Span	Joan, you won't believe it - I performed the lead role of my hip hop piece last night and it rocked!

Figure 9: An example temporal QA for Temporal Precedence

Question: How many months after hiking with his dad did Sam take his friends on an epic hiking trip?
Options: A. About 7 months; B. About 8 months; C. About 6 months; D. About 10 months; **E. Unanswerable**

Event	Sam went hiking with his dad
Person	Sam
Relative Time	when I was ten
Session ID	session_0
Session Time	05/18/2023
Text Span	I did this hike with my dad way back when I was ten.

Event	Sam took his friends on a hiking trip
Person	Sam
Relative Time	last Friday
Session ID	session_21
Session Time	12/31/2023
Text Span	I'm really getting into this healthier lifestyle, just took my friends on an epic hiking trip last Friday!

Figure 10: An example temporal QA for Temporal Interval

You will be shown a <NUM>-line conversation between two speakers. Please read, memorize and understand the conversation, then complete timeline summarization under the guidance of Task Introduction.

Task Introduction
1 - Generate a summary in a paragraph based on the whole conversation for the given timestamp. In the summary, you should focus on all what speakers mentioned in the conversation and express in a way that what the speaker mentioned.
2 - For each mentioned event in the conversation that speakers did at different timestamps than the given timestamp, also generate a summary about it from the perspective of its corresponding timestamps.
3 - Determine the chat range of each summary based on line numbers.
4 - For each generated summary, including the summary for the given conversation and all mentioned events, link them to their corresponding inferred time. The timestamp should be in the format of month/day/year if the exact date can be obtained. If only rough time can be inferred, such as "last week", give the inferred time range without expressing with relative time. For example, given the current timestamp "08/03/2022" and a speaker mentioned a past event happened "last week", you should link its summary to "During 07/24/2022 to 07/30/2022". Another example is that given the timestamp "02/21/2022", a mentioned event for "next year" should be linked to "During 2023".
5 - Report time and summary in JSON format only with the assigned keys: "time", "summary". For example, assuming the conversation at 08/03/2022 talks about Angela won another award for her cold cure last week from line 1 to line N. Thus, its task result could be: [{"time": "During 07/24/2022 to 07/30/2022", "summary": "Angela won another award for her cold cure", "start": 1, "end": N}, {"time": "08/03/2022", "summary": "Angela shares the exciting news that she won another award for her cold cure last week", "start": 1, "end": N}].

Figure 11: Prompt for Time-aware Memorization.

You will be shown 1 Query Question and <NUM> Memory Options. Please read, memorize and understand given materials, then complete the task under the guidance of Task Introduction.

Query Question:
<QUERY>
Memory Options:
<MEMORY>

Select one or more topics from Topic Options that relevant with Query Question and output in JSON format with a key: "selected topics". Do not report the option content, but only report selected option numbers in a string separated with "#". For example, if memory options N and M are chosen, then the output is {"selected topics": "N#M"}. For Query Question in the task, any chosen option numbers should be larger than 0 but not exceed the total number of Memory Options <NUM>.

Figure 12: Prompt for memory retrieval.

Given related evidence and a question with answer options, your task is to parse them into python code based on python packages 'datetime' and 'dateutil'. Note that in 'dateutil', it defines abbreviations for days of weeks: MO (Monday), TU (Tuesday), WE (Wednesday), TH (Thursday), FR (Friday), SA (Saturday) and SU (Sunday). Additionally, we have implemented some functions in 'temporal_utils.py' to support you:
- 'weekRange(query_date)' returns a tuple of dates, 'start_date' and 'end_date', which are the start date and the end date for the week that 'query_date' is in.
- 'lastWeekendRange(query_date)' returns a tuple of dates, 'start_date' and 'end_date', which are the start date and the end date for the last weekend of 'query_date'.
As the question is about temporal reasoning, your parsed code should cover defining variables for timestamps and calculations over time values. Make sure to distinguish the timestamps for when a person mentioned an event and when the person did the event. And you should define the timestamp of the event that time can't be inferred as None, and the question should be unanswerable. Your output should be in JSON format with only one key 'program' with the generated code in String as the value, such as {'program': 'from datetime import date'} Below are some examples:
<EXAMPLES>

Related Evidence:
<MEMORY>
Question:
<QUERY>
Answer Options:
<OPTIONS>

Figure 13: Prompt for generating Python code for temporal reasoning.

You are an intelligent dialog bot. You will be shown Related Evidence supporting for Query Question and each turn in Related Dialogue has been marked with its timestamp. Additionally, you are also given Rationales which are python code with its output for supporting the reasoning and answering the question. Please read, memorize, and understand given materials, then select the most suitable and factually correct options step by step from the given Answer Options. And you should consider the "Unanswerable" option if the answer can't be obtained. Your output should be in JSON format with only one key "final answer" with values from the five letters (A/B/C/D/E), such as {"final answer": "..."}.

Related Evidence:
<MEMORY>
Query Question:
<QUERY>
Answer Options:
<OPTIONS>
Rationales:
<CODE>

Figure 14: Prompt for temporal question answering.