Updating Large Language Models' Memories with Time Constraints

Xin Wu[1,2], Yuqi Bu[1,2], Yi Cai[1,2,*], Tao Wang[3],
[1]School of Software Engineering, South China University of Technology
[2]Key Laboratory of Big Data and Intelligent Robot
(South China University of Technology) Ministry of Education
[3]Department of Biostatistics and Health Informatics, King's College London
***Correspondence:** ycai@scut.edu.cn

Abstract

By incorporating the latest external knowledge, large language models (LLMs) can modify their internal memory. However, in practical applications, LLMs may encounter outdated information, necessitating the filtering of such data and updating of knowledge beyond internal memory. This paper explores whether LLMs can selectively update their memories based on the time constraints between internal memory and external knowledge. We evaluate existing LLMs using three types of data that exhibit different time constraints. Our experimental results reveal the challenges most LLMs face with time-constrained knowledge and highlight the differences in how various LLMs handle such information. Additionally, to address the difficulties LLMs encounter in understanding time constraints, we propose a two-stage decoupling framework that separates the identification and computation of time constraint into a symbolic system. Experimental results demonstrate that the proposed framework yields an improvement of over 60% in ChatGPT's performance, and achieves a 12-24% enhancement in state-of-the-art LLM GPT-4.

1 Introduction

Large language models (LLMs) can provide answers that differ from their internal knowledge when supplied with external information (Zheng et al., 2023; Xie et al., 2024). This capability facilitates the updating of incorrect or outdated information within LLMs (Yao et al., 2023). For instance, as illustrated in Figure 1(a), when asked "Who is the Prime Minister of the UK?", the LLM can provide the updated answer "Rishi Sunak," based on the latest knowledge, instead of relying on its outdated memory like "Boris Johnson."

In practical applications such as retrieval-augmented tasks (Gao et al., 2023b), outdated information may also be inputted into LLMs alongside the latest knowledge. This necessitates LLMs to eliminate knowledge overlapping with their memory and update their internal memory with new information. A straightforward approach involves selectively updating the internal memory of LLMs based on the time constraint relationship between the question, internal memory, and the external knowledge. For example in Figure 1(b), given the question "How many LA metro rail stations as of 2024", the LLM's original internal memory states "as of 2021, there are 93." The first given piece of knowledge has a time constraint of "May 20, 2016", which is covered by the LLM's current internal memory range, thus no update is needed. The second piece of knowledge has a time constraint of "2022-2024", which extends beyond the LLM's internal memory and necessitates an update. Ignoring these time constraints can lead LLMs to provide answers that either exceed the scope of the intended question constraints or result in redundant updates to already covered knowledge.

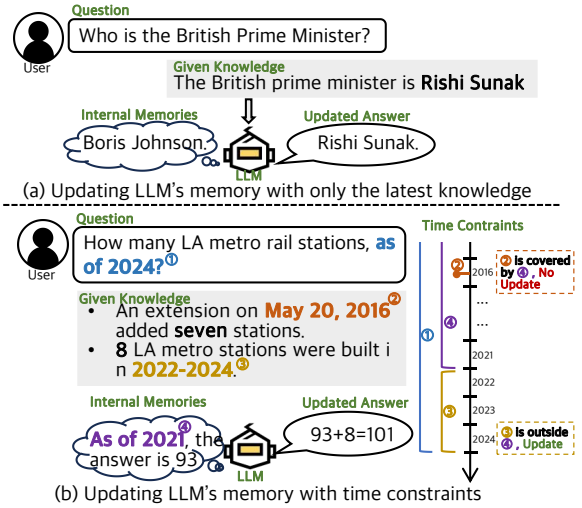

Figure 1: Updating LLM's memory without and with time constraints.

This paper conducts a comprehensive and controlled study of how LLMs understand and update

Findings of the Association for Computational Linguistics: EACL 2024, pages 13693–13702

time-constrained knowledge. Specifically, we construct test data based on three time constraint relations between internal memory and external knowledge (*i.e.*, complete inclusion, intersection, and disjunction) to examine whether LLMs can comprehend and selectively update time-constrained knowledge. Additionally, we investigate the sensitivity of LLMs to the sequential order of time-constrained knowledge and the role of internal memory in the process of knowledge updating. Our experiments reveal several key findings:

(1) **Most LLMs struggle to update memory with time-constraint.** Except for GPT4, the other LLMs (*e.g.*, Llama2, Mistral) exhibit poor abilities for updating memory with time constraints. Additionally, ChatGPT performs worse in this regard compared to smaller open-source models across multiple scenarios.

(2) **Different LLMs display distinct decision styles when handling time-constrained knowledge.** GPT4 is more proactive in accepting time-constrained knowledge, while ChatGPT generally depends on its internal memory.

(3) **Time constraints can affect LLMs' decision styles.** Time constraints in the question act as triggers to promote the stubborn LLMs, *e.g.*, ChatGPT, to update their internal memory more frequently.

In contrast to the poor temporal reasoning of LLMs (Zhu et al., 2023; Li et al., 2023), the discovery of the the relationship of time constraints are easily achievable in symbolic systems (Li et al., 2023). For example, in a Python solver, representing time-related information using specific data structures enables checking for temporal contradictions among multiple pieces of information. On the other hand, LLMs have shown strong capabilities in information extraction (Zhu et al., 2023; Pan et al., 2023). Based on this, we propose a two-stage decoupling framework for time-constrained knowledge updating. We limit LLMs to information extraction only and delegate information reasoning to symbolic systems (*i.e.*, Python solver). We conduct experiments on Mistral, ChatGPT, and GPT4, and the results demonstrate significant performance improvements for these three LLMs.

2 Related Work

Knowledge Editing. As the global landscape changes, there is a growing need to update LLMs without retraining them entirely. Knowledge editing has emerged as a solution to this challenge (Meng et al., 2022a,b; Mitchell et al., 2022; Zheng et al., 2023; Li et al., 2024). The current knowledge editing methods can be mainly divided into two categories. The first involves modifying the parameters of LLMs to alter their internal knowledge (Meng et al., 2022a,b; Li et al., 2024). The second approach injects knowledge directly into the prompt, leveraging in-context learning to update the LLMs (Mitchell et al., 2022; Zheng et al., 2023; Xie et al., 2024). A comprehensive review with unified experiments suggests that the latter method more effectively ensures the reliability, generalization, and locality of knowledge updates (Yao et al., 2023). This paper primarily investigates the capacity of LLMs to perform knowledge editing within time constraints.

Time-sensitive Question Answering. Time-sensitive QA is closely related to knowledge updating. Since only knowledge that changes over time needs updating. Currently, there is a growing body of work on benchmarking LLMS' temporal reasoning capabilities (Chen et al., 2021; Wei et al., 2023; Vu et al., 2023; Tan et al., 2023). However, most time-sensitive QA datasets measure whether LLMs can obtain answers to preset questions based on contexts with time constraints. This differs from knowledge updating, which modifies the internal memory of LLMs. Section 3.1 discusses how to construct knowledge with time constraint from some existing time-sensitve QA datasets.

Symbolic system Enhancement. LLMs often struggle with rigorous logical reasoning and numerical calculations (Liu et al., 2023; Pan et al., 2023). Combining LLMs with symbolic systems has proven effective in mitigating these challenges (Pan et al., 2023; Li et al., 2023; Zhu et al., 2023). Inspired by this, we propose a symbolic-enhanced framework to update LLMs' memories with time constraints. Our approach involves identifying time constraint relationships using Python code and performing reasoning with a Python solver.

3 Dataset Construction

We introduce a framework to construct time-constrained knowledge. Following prior work (Mitchell et al., 2022; Zheng et al., 2023; Xie et al., 2024), we adopt question answering (QA) task as the testbed for evaluation. First, we collect questions whose answers may change over time. Second, we elicit LLMs' internal memories about these questions. Finally, we construct new time-

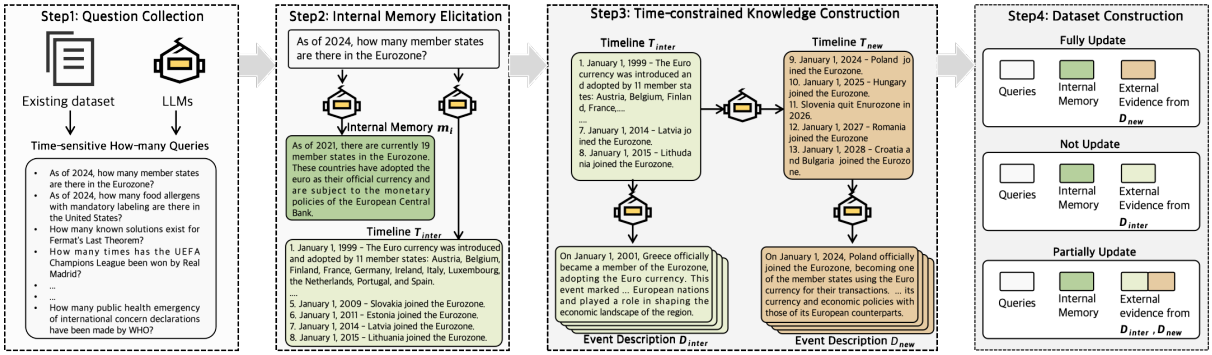

Figure 2: Our framework to construct dataset with time-constrained knowledge.

constrained knowledge that need to be updated into LLMs' memories.

3.1 Task Definition

Given a question q_i and supporting evidence e_i, requiring LLMs to answer q_i based on both e_i and its internal memories m_i, and obtain the answer a_i. Our experiment focuses on testing whether LLMs can understand and process time constraints in e_i and m_i.

To construct data with time constraints in e_i and m_i, in this paper, we mainly use **time sensitive how-many** questions as q_i. Since the answer to time-sensitive how-many questions usually is a number that changes over time, we can build a lot of external evidence e_i to indicate that the answer is changing. LLMs need to analyze the time constraints between external evidence e_i and internal memories m_i and update their memories. For example, the answer to the question "How many subway stations are there in Los Angeles?" will continually increase or decrease with the addition of new evidence.

3.2 Questions Collection

According to the task definition, we limit the scope of the questions to **time-sensitive how-many** types of questions. For example, "In 2024, how many subway stations are there in LA?" As illustrated in Figure 2 step 1, we construct time-sensitive how-many questions Q in two ways. Firstly, we extract them from existing time-sensitive or how-many QA datasets. Secondly, to enhance question diversity, we utilize ChatGPT to generate questions spanning various domains such as movies, sports, literature, and politics.

3.3 Internal Memories Elicitation

As shown in Figure 2 step 2, we obtain internal memories m_i from LLMs in a closed-book manner. We instruct LLMs to answer questions q_i without any external evidence. For example, given a question "how many subway stations are there in LA?", LLMs are instructed to provide an answer "93" and its supporting background information.

To facilitate the next step of time-constrained knowledge construction, we also convert internal memories into a timeline T_{inter}. As shown in Figure 2, we guide the LLMs to construct a timeline correlating with the question. To ensure the stability of testing, we manually filter out samples where the answer, internal memories, and timeline are inconsistent.

3.4 Time-Constrained Knowledge Construction

Given the inconsistent cutoff times for training data across various LLMs are not consistent, but with all cutoffs occurring before 2024, we define "new knowledge" as events fabricated to occur after 2024. To achieve this goal, as illustrate in Figure 2 step 3, we instruct the LLMs to extend the internal memory timeline T_{inter} to obtain a new timeline $T_{all} = T_{inter} + T_{new}$, where T_{new} is the timeline after 2024. For each event in T_{inter} and T_{new}, we instruct LLMs to paraphrase them to include more details. Thus, we obtain sets of event descriptions D_{inter} and D_{new}.

3.5 Dataset Construction

There are various time constraint relationships between internal memory and external evidence, such as complete inclusion, intersection, and disjunction. Based on these relationships, we constructed three types of data to assess the performance of LLMs

	w/o Internal memories			w/ Internal memories		
	Fully	Partially	Not	Fully	Partially	Not
			BASE			
Llama2 (7b-chat)	7.76	16.81	51.54	25.43	18.97	42.05
Vicuna (7b-v1.5)	11.21	14.22	47.16	19.83	16.38	55.68
Mistral (7b-instruct-v0.2)	11.64	19.83	47.73	47.84	31.47	51.7
ChatGPT (gpt3.5-turbo)	6.47	9.05	**63.64**	19.83	8.62	68.75
GPT4 (gpt4-turbo)	**44.4**	**44.4**	54.55	**81.03**	**60.78**	**69.32**
			COT			
Llama2 (7b-chat)	8.62	15.09	39.77	28.88	18.1	49.43
Vicuna (7b-v1.5)	9.48	16.81	40.34	31.9	18.97	56.82
Mistral (7b-instruct-v0.2)	17.24	17.67	34.09	51.72	25.43	56.82
ChatGPT (gpt3.5-turbo)	12.93	23.71	53.41	36.64	15.95	67.05
GPT4 (gpt4-turbo)	**57.33**	**51.29**	**68.18**	**84.48**	**73.71**	**76.14**

Table 1: The accuracy of tested LLMs on the construted dataset.

in different scenarios. As shown in Figure 2 step 4: **Fully Update**, **Not Update**, and **Partially Update**. Each sample x_i of is defined as $\{q_i, m_i, e_i, a_i\}$, where m_i and a_i represent the internal memories and answer corresponding to question q_i. The difference among the three types of data lies in the use of different external supporting evidence e_i. Specifically:

Fully Update: The e_i used for Fully update type data is entirely extracted from D_{new}. In this case, all information mentioned in e_i needs to be updated into the answer. This type of data mainly verifies whether LLMs have basic capabilities to identify and update time-constrained knowledge.

Not Update: The e_i used for data of the Not update type comes entirely from D_{inter}. In this case, all the information mentioned in e_i does not need to be updated into the answer because this information has already been mentioned in the internal memories. This type of data mainly verifies whether LLMs have the ability to discern whether the accepted information overlaps with their internal memories.

Partially Update: The e_i used for Partially Update type of data comes from D_{inter} and D_{new}. In this case, the information mentioned in e_i that belongs to D_{new} needs to be updated into the answer, while the information belonging to D_{inter} does not need to be updated. Partially updated data is mainly for verifying whether LLMs can selectively process the received information.

After manually filtering out inconsistent data, we ultimately collect 640 test samples for each LLMs, with 232 Fully Update samples, 176 Not Update samples, and 232 Partially Update samples.

4 Experiments

4.1 Dataset and Metric

We conduct experiments using the constructed dataset. Specifically, for each LLM, the test data consists of 640 samples with their internal memories, *i.e.*, $\{q_i, m_i, e_i, a_i\}$, and 640 samples without internal memories, *i.e.*, $\{q_i, e_i, a_i\}$. This setup aims to determine the ability of LLMs to accurately recall their internal memories during the knowledge update process. Throughout all experiments, we employ accuracy as the metric.

4.2 Baselines

We select widely used LLMs and prompt methods as baselines. Specifically, we experiment with ChatGPT (Ouyang et al., 2022), GPT4 (Achiam et al., 2023), Llama2 (Touvron et al., 2023), Vicuna (Chiang et al., 2023), and Mistral (Jiang et al., 2023) as LLMs, and conduct experiments using two prompt methods: BASE and COT. Specifically, BASE prompt directly prompts the LLM to generate the answer to the question, while COT prompt involves breaking down the question and gradually obtaining the answer (Kojima et al., 2022).

4.3 Experimental Setup

For the BASE method, we directly use the responses generated by LLMs as the predicted answers. For the COT method, we take an additional step to extract the final answer from the responses of LLMs. We employ ChatGPT to perform this extraction step.

		Fully	△	Partially	△	Not	△
With time constraint in question							
ChatGPT (gpt3.5-turbo)	BASE	19.83	-	8.62	-	68.75	-
	COT	36.64	-	15.95	-	67.05	-
Mistral (7b-instruct-v0.2)	BASE	47.84	-	31.47	-	51.70	-
	COT	51.72	-	25.43	-	56.82	-
Without time constraint in question							
ChatGPT (gpt3.5-turbo)	BASE	3.88	-15.95	1.72	-6.90	99.43	+30.68
	COT	20.26	-16.38	8.62	-7.33	97.73	+30.68
Mistral (7b-instruct-v0.2)	BASE	34.91	-12.93	22.41	-9.06	83.52	+31.82
	COT	31.03	-20.69	13.79	-11.64	79.55	+22.73

Table 2: Comparison analysis of whether the question contains time constraints. △ denotes the change in performance after removing the time constraint in question.

4.4 Results

Most LLMs struggle to update memories with time constraints. As shown in Table 1, small LLMs with 7b parameters (Llama2, Vicuna, Mistral) and ChatGPT perform poorly across three data types. Notably, ChatGPT, using the BASE method, often underperforms compared to the 7 billion parameter models. Among current models, only GPT-4 shows relatively good proficiency in updating time-constrained knowledge, yet it still fails to achieve an accuracy of 70% in many cases.

Some LLMs are stubborn, while some LLMs are proactive. LLMs exhibit certain preferences across three types of data. For instance, GPT-4 generally performs optimally but has a notable drop in performance with Not Update data compared to its robust outcomes with Fully Update and Partially Update data. This suggests that GPT-4 excels at incorporating new information but struggles in scenarios where updates are not required, *i.e.*, Not Update. Conversely, although ChatGPT does not perform as well as some 7b LLMs on Fully Update and Partially Update data, it excels on Not Update data, sometimes even outperforming GPT-4. This indicates ChatGPT's more conservative strategy that it rely on its own memory.

LLMs exhibit amnesia phenomenon. Our experiments reveal significant performance differences in LLMs depending on whether internal memories are explicitly present (w/ Internal memories) or absent (w/o Internal memories). This indicates that even if LLMs possess correct memories, the absence of explicit memory retrieval can lead to memory loss during response generation. These findings corroborate existing research (Xie et al., 2024) and highlight the importance of explicitly informing LLMs of known knowledge to enhance their performance.

COT facilitates knowledge updating for large LLMs, but not small LLMs. We can observe that the COT method brings positive improvements to LLMs such as ChatGPT and GPT4 (except for a decline in COT+ChatGPT on Not Update). However, COT does not consistently improve small-parameter models like Llama2, Vicuna, and Mistral. One possible reason is ChatGPT and GPT4 possess stronger chain reasoning capabilities.

The time constraint in the question is the trigger for changing the LLMs' decision styles. Apart from the time constraints in knowledge, we also compare the impact of time constraints in question q_i. Table 2 presents the experimental results for the open-source LLM Mistral and the closed-source ChatGPT. We make the following observations: (1) When there is no time constraint in the question, the performance of both Mistral and ChatGPT decrease on both Fully Update and Partially Update data types. This indicates that these two LLMs are more inclined to rely on their own memory in such cases, showing that the presence or absence of a time constraint in the question can change the decision styles of the LLMs. This also provides engineering insight, indicating that even when useful knowledge is provided to LLMs, explicit time constraints in the question are needed to trigger LLMs to update their memories. (2) On the Not Update dataset, when there are no time constraints in the question, the LLMs tend not to update. Therefore, the accuracy of both Mistral and ChatGPT significantly improved.

LLMs are sensitive to the order of information. The input internal memories and external evidences

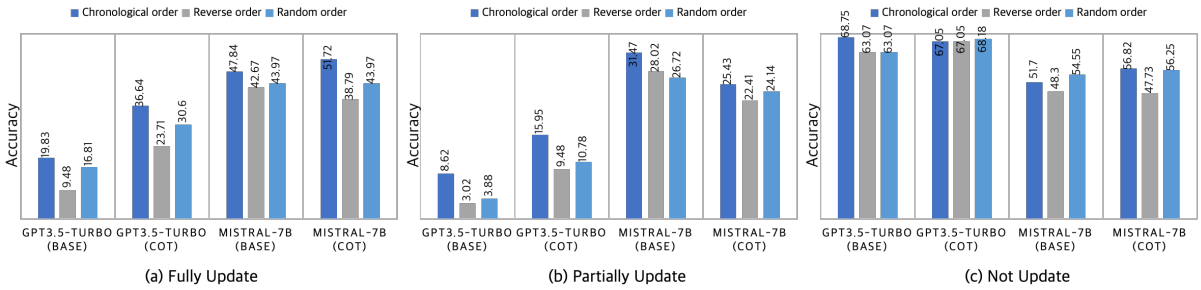

Figure 3: The comparative analysis of the order of information in the instructions.

	Fully	△	Partially	△	Not	△
	PAL					
Mistral (7b-instruct-v0.2)	43.53	-4.31	30.17	-1.3	65.91	+14.21
ChatGPT (gpt3.5-turbo)	67.67	+47.84	34.91	+26.29	63.64	-5.11
GPT4 (gpt4-turbo)	90.09	+9.06	**81.9**	+21.12	91.48	+22.16
	Two-stage Decoupling Framework					
Mistral (7b-instruct-v0.2)	65.52	+17.68	48.71	+17.24	81.82	+30.12
ChatGPT (gpt3.5-turbo)	77.59	**+57.76**	75.43	**+66.81**	**94.89**	**+26.14**
GPT4 (gpt4-tubo)	**93.97**	+12.94	80.6	+19.82	93.75	+24.43

Table 3: The accuracy of code-enhanced methods. △ indicates performance gain over BASE method.

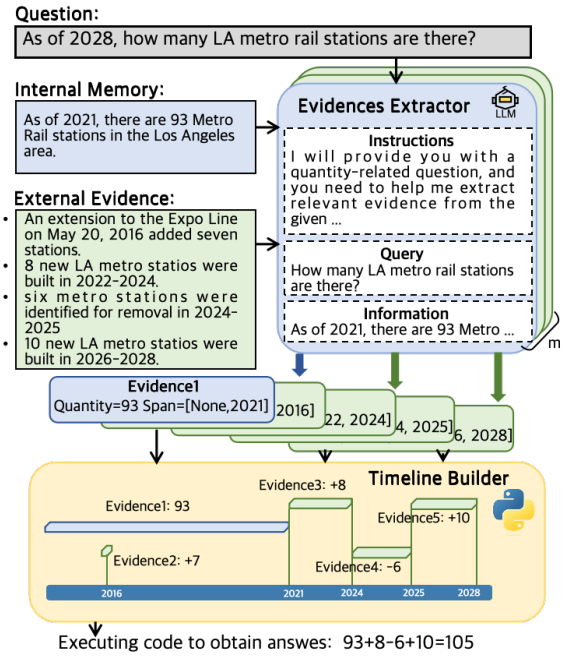

Figure 4: Two-stage decoupling framework.

in each sample are arranged in chronological order. An intuitive question is whether LLMs are sensitive to the order of received information. To investigate this, we conduct a set of comparative experiments testing three order types: chronological, reverse, and random. Figure 3 shows the results of Mistral and ChatGPT on these three types of ordered data. Our findings indicate that both Mistral and ChatGPT exhibit varying degrees of sensitivity to order. Performance notably decrease with reverse order across all data types. For random order, performance is intermediate, falling between the results for chronological and reverse orders.

5 Two-stage Decoupling Framework

The above experimental results indicate that, except for GPT-4, most LLMs exhibit weak abilities to identify and update time-constrained knowledge. One possible reason is that LLMs lack of understanding of temporal reasoning (Wei et al., 2023). In contrast, LLMs demonstrate excellent performance in information extraction (Zhu et al., 2023; Pan et al., 2023). Additionally, most temporal reasoning problems can be solved through symbolic computation (Gao et al., 2023a), such as a Python solver. Therefore, we propose a two-stage framework for updating time-constrained knowledge. We decouple information extraction and temporal reasoning into two separate modules. Specifically, this framework includes an Evidence Extractor for extracting time-sensitive information and a Timeline Builder for calculating the final answer based on the extracted information. The overall architecture is illustrated in Figure 4.

5.1 Evidence Extractor

Formally, Evidence Extractor is a LLM M that takes question q_i, internal memories m_i, and external information e_i as input. It extracts a set of evidence E from m_i and e_i that is necessary to answer q_i. All evidence is represented in the form of Python class. Specifically, each evidence consists of a quantity, a start time and a end time. For example, the information "2 new LA metro stations were built in 2023-2024." would be extracted as evidence: "evidence1 = Evidence(quantity=2, start_time=2023, end_time=2024)."

5.2 Timeline Builder

Formally, Timeline builder is a task-specific Python solver P. We utilize the logical reasoning mechanism within Python to construct a timeline from the extracted evidence set E. For any time constraint t_i in question, P is able to return the evidence subset prior to t_i along with the total sum of quantities corresponding to all evidence in the subset.

5.3 Framework Evaluation

We validate the effectiveness of the proposed framework on the constructed dataset. Specifically, since the framework requires explicit input of internal memories, we test it on data with internal memories (w/ Internal memories). We select three LLMs: Mistral, ChatGPT, and GPT-4, based on preliminary experiments that demonstrate their code comprehension and generation capabilities. Additionally, we chose the code enhanced prompt method PAL as the baseline (Gao et al., 2023a). The experimental results are shown in Table 3. We make the following observations:

Code-enhanced methods facilitate updating time-constrained knowledge. The experimental results indicate that the PAL method generally improves performance compared to the BASE method. For instance, with GPT-4, PAL achieves improvements of 9.06%, 21.12%, and 22.16% across three different data types. In the case of ChatGPT, PAL significantly enhances performance by 47.84% on Fully Update and 26.29% on Partially Update datasets. This improvement may be attributed to PAL's ability to adjust ChatGPT's conservative strategy in the BASE method, steering it towards more frequent knowledge updates. However, this adaptation leads to a performance decline on Not Update data, where updates are unnecessary. Conversely, for Mistral, a model with weaker coding abilities, PAL results in performance decreases of 4.31% and 1.30% on Fully Update and Partially Update datasets, respectively. These outcomes suggest that the effectiveness of PAL is influenced by the coding capabilities of the LLMs.

Decoupling is beneficial for identifying and updating time-constrained knowledge. The proposed two-stage framework consistently yields significant improvements across all cases. For instance, it enhances ChatGPT's performance by 57.76% on Fully Update and 66.81% on Partially Update tasks. Even the top-performing GPT-4 model shows improvements of 12.94%, 19.82%, and 24.43% across three types of data. Furthermore, by decoupling the framework, LLMs can focus on information extraction, reducing errors associated with code generation compared to PAL, thereby providing stable enhancements across all tested LLMs.

LLMs are good at information extraction. We manually verify the evidence extraction capabilities of Mistral, ChatGPT, and GPT-4 due to the absence of ground truth. Specifically, for each LLM, we randomly sample 50 instances. The accuracy of evidence extraction for the three LLMs is 68.00%, 80.00%, and 88.00%, respectively. This indicates that LLMs can perform the extraction task well in most cases. We further analyze the errors, which predominantly fall into three categories: semantic understanding errors, instruction following errors, and code generation errors.

- Semantic understanding errors involve the LLMs failing to comprehend the information correctly, either generating evidence not mentioned or omitting relevant evidence. These issues, often due to hallucinations, represent 10.00%, 12.00%, and 8.00% of errors for Mistral, ChatGPT, and GPT-4, respectively. For example, for information ""Breaking Bad" had a total of 62 episodes spread across 5 seasons as of the latest information.", LLM incorrectly takes the number of episodes 62 in the text as the number of seasons.

- Instruction following errors occur when the LLMs misinterpret the extraction instructions as a question-answering task and generate the final answer directly rather than extracting the required evidence. These errors primarily affect Mistral and ChatGPT, accounting for 10% and 4%, respectively. For example, for information "The FDA added sesame as the

	Case 1	Case 2
Question	As of 2028, how many member states are there in the Eurozone?	As of 2030, how many food allergens with mandatory labeling are there in the United States?
Answer	20	14
Internal Memory	As of 2021, there are currently 19 member states in the Eurozone....	As of 2021, there are eight food allergens with mandatory labeling in the United States...
External Evidence	On January 1, 2024, Poland officially adopted the Euro currency and became a member of the Eurozone....	1.In 2024, the FDA made an important update ... by adding sesame to the list of mandatory allergens for labeling. 2.In 2026, the FDA made an significant update ... by adding molluscan shellfish and mustard .... 3.In 2028, FDA included Shellfruit as a mandatory allergen for labeling purposes. 4.In 2029, the FDA expanded the list ... by adding sunflower seeds and quinoa protein.
BASE	19	12
COT	Sure, let's break it down step by step. First, we know that as of 2021, there are 19 member states in the Eurozone. Next, we need to determine how many years have passed between 2021 and 2028. 2028 - 2021 = 7 years So, if we assume that no countries have joined or left the Eurozone during those 7 years, then as of 2028, there would still be 19 member states in the Eurozone.	Step 1: We know that as of 2021, there are eight food allergens with mandatory labeling in the United States. Step 2: We also know that there have been several updates to the list of mandatory allergens for labeling in the United States, with sesame, molluscan shellfish, mustard, and Shellfruit being added in 2024, 2026, 2028, and 2029, respectively. Step 3: Based on the information provided, we can add these four allergens to the original list of eight, bringing the total number of mandatory allergens for labeling to 12 .
Ours	Evidence1 = Evidence(19, year=2021) Evidence2 = Evidence(1, year=2024, month=1, day=1) Answer = 19+1 = 20	Evidence1 = Evidence(20, year=2021) Evidence2 = Evidence(1, year=2024) Evidence3 = Evidence(2, year=2026) Evidence4 = Evidence(1, year=2028) Evidence5 = Evidence(2, year=2029) Answer = 8+1+2+1+2 = 14

Table 4: Comparison of generation results under different methods used by ChatGPT. The baseline methods generate incorrect responses, while our proposed framework can provide correct answers.

9th mandatory allergen for labeling.", LLM mistakenly thought to answer the question and extract 9 as the evidence.

- Code generation errors mainly involve the generation of incorrect or redundant code by the LLMs. The proportions of code generation errors for the three LLMs are 12.00%, 4.00%, and 4.00%, respectively.

Case Study Figure 4 depicts ChatGPT's cases under different methods. In the first scenario, ChatGPT does not identify any time constraint conflicts between the given knowledge and its memory. Therefore, ChatGPT does not update its internal memory. Conversely, in the two-stage framework, ChatGPT accurately extracts temporal information from both its internal memory and the provided knowledge, updating it precisely with the help of a Python solver. In the second scenario, although ChatGPT detects time constraint contradictions using the BASE and COT methods and try to update its memory. However, a computational error occurs during the update. In the proposed framework, calculations are performed by a Python solver, ensuring no computational errors. These examples demonstrate that while LLMs can accurately extract time-related information, they cannot independently perform temporal reasoning accurately. In contrast, a Python solver can effectively execute temporal reasoning. The proposed framework effectively combines the strengths of LLMs and symbolic systems in information extraction and temporal reasoning, respectively.

Generalization Ability Our proposed method can be transfer to non-numerical case. The core idea of the decoupling framework is to decouple temporal reasoning into two steps, completed by the LLM and a symbolic system, respectively. The experiment results show that the LLM's ability to extract information is far superior to its reasoning ability. Based on this conclusion, we can quickly transfer this two-stage framework to other tasks. We supplement our experiments with time-sensitive entity type QA data (Wei et al., 2023). We utilize LLM to extract time-sensitive evidence from the context and answer questions by matching the timing with

	Accuracy
BASE	40.81
COT	42.85
Our Two-stage Framework	**57.14**

Table 5: Accuracy on entity-type QA task.

the question. For example, one piece of knowledge is like "Detmer was traded to the Cleveland Browns in 1999 ; the Browns wanted him to mentor rookie quarterback Tim Couch . Detmer started the first game of the 1999 season , then served as backup until Couch sprained his foot in week 15 . He started the final game of the 1999 season . Detmer injured his right Achilles and was inactive the entire 2000 season ." LLM extract evidence as: "{"subject": "Ty Detmer", "relation": "team", "object": "Cleveland Browns", "start_time": 1999, "end_time": 2000}", which can be used to answer the question"Ty Detmer played for which team from 1999 to 2000?" The results using different methods in ChatGPT are as showed in Table 5. This demonstrates that our method has good generalizability across different scenarios.

6 Conclusion

In this paper, we explore the capabilities of LLMs for updating knowledge with time constraints. We propose a framework for constructing test data for time-constrained knowledge and construct test data to evaluate LLMs including Llama2, Vicuna, Mistral, ChatGPT, GPT4, respectively. Experimental results reveal that most LLMs lack the ability to update time-constrained knowledge. On the other hand, different models also exhibit different decision styles for time-constrained knowledge. Additionally, we find that time constraints in question is a trigger that can change the LLMs' decision styles. Finally, we propose a two-stage decoupling framework that decouples the discovery and computation of time constraints from LLMs into a symbolic system. Experimental results show that this decoupling approach can bring ChatGPT an improvement of over 60%, and also has a 12-24% improvement for state-of-the-art LLMs GPT4.

Acknowledgements

This research is supported by the National Natural Science Foundation of China (62076100, 62476097), the Fundamental Research Funds for the Central Universities, South China University of Technology (x2rjD2240100), the Science and Technology Planning Project of Guangdong Province (2020B0101100002), Guangdong Provincial Fund for Basic and Applied Basic Research—Regional Joint Fund Project (Key Project) (2023B1515120078), Guangdong Provincial Natural Science Foundation for Outstanding Youth Team Project (2024B1515040010), the China Computer Federation (CCF)-Zhipu AI Large Model Fund. TW was funded by a Early Career Research Award from the Institute of Psychiatry, Psychology & Neuroscience; Maudsley Charity, and the National Institute for Health Research (NIHR) Biomedical Research Centre at South London and Maudsley NHS Foundation Trust and King's College London.

Limitations

In this paper, we explore the three common types of time constraints in our experiments: complete inclusion, intersection, and disjunction. However, time constraints in real-world scenarios can be more complex and may require external knowledge retrieval to aid reasoning. Additionally, we primarily examine the ability of LLMs to update knowledge through question-answering (QA) tasks, as well as considering their performance in other tasks such as natrual language generation task.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

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)*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023a. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023b. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. 2024. Pmet: Precise model editing in a transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18564–18572.

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 preprint arXiv:2305.15014*.

Hanmeng Liu, Ruoxi Ning, Zhiyang Teng, Jian Liu, Qiji Zhou, and Yue Zhang. 2023. Evaluating the logical reasoning ability of chatgpt and gpt-4. *arXiv preprint arXiv:2304.03439*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022a. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372.

Kevin Meng, Arnab Sen Sharma, Alex J Andonian, Yonatan Belinkov, and David Bau. 2022b. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. 2022. Memory-based model editing at scale. In *International Conference on Machine Learning*, pages 15817–15831. PMLR.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. 2023. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

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.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Tu Vu, Mohit Iyyer, Xuezhi Wang, Noah Constant, Jerry Wei, Jason Wei, Chris Tar, Yun-Hsuan Sung, Denny Zhou, Quoc Le, et al. 2023. Freshllms: Refreshing large language models with search engine augmentation. *arXiv preprint arXiv:2310.03214*.

Yifan Wei, Yisong Su, Huanhuan Ma, Xiaoyan Yu, Fangyu Lei, Yuanzhe Zhang, Jun Zhao, and Kang Liu. 2023. Menatqa: A new dataset for testing the temporal comprehension and reasoning abilities of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1434–1447.

Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. 2024. Adaptive chameleon or stubborn sloth: Revealing the behavior of large language models in knowledge conflicts. In *The Twelfth International Conference on Learning Representations*.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing large language models: Problems, methods, and opportunities. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. 2023. Can we edit factual knowledge by in-context learning? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4862–4876.

Xinyu Zhu, Cheng Yang, Bei Chen, Siheng Li, Jian-Guang Lou, and Yujiu Yang. 2023. Question answering as programming for solving time-sensitive questions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12775–12790.