Measuring Iterative Temporal Reasoning with Time Puzzles

Zhengxiang Wang*
Department of Linguistics & IACS
Stony Brook University
zhengxiang.wang@stonybrook.edu

Zeyu Dong*
Department of Applied Math and Statistics
Stony Brook University
zeyu.dong@stonybrook.edu

Abstract

Tool use, such as web search, has become a standard capability even in freely available large language models (LLMs). However, existing benchmarks evaluate temporal reasoning mainly in static, non-tool-using settings, which poorly reflect how LLMs perform temporal reasoning in practice. We introduce Time Puzzles, a constraint-based date inference task for evaluating *iterative temporal reasoning with tools*. Each puzzle combines factual temporal anchors with (cross-cultural) calendar relations and may admit one or multiple valid dates. The puzzles are algorithmically generated, enabling controlled and continual evaluation. Across 13 LLMs, even the best model (GPT-5) achieves only 55.3% accuracy without tools, despite using easily searchable facts. While web search improves performance, models perform substantially better when constraints are rewritten with explicit dates, removing the need for factual lookup. These results reveal a gap in reliable tool use for iterative temporal reasoning.

1 Introduction

Modern LLMs are increasingly deployed with external tools, such as web search and code interpreter, even in free customer-facing settings (OpenRouter.ai, 2025). These capabilities allow models to retrieve factual information and verify intermediate reasoning steps during inference. However, existing temporal reasoning benchmarks largely evaluate models in *static, single-shot* settings, which do not reflect the iterative, tool-augmented workflows increasingly used in practice.

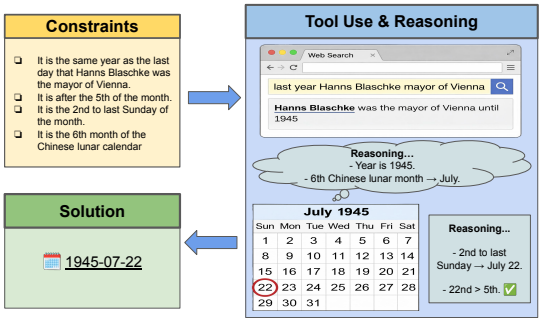


Figure 1: Time Puzzles: a simple date inference task requiring iterative tool-aided temporal reasoning.

Prior work on temporal reasoning has focused on static, single-shot settings (Ning et al., 2020; Zhou et al., 2021, 2019; Qin et al., 2021). While recent benchmarks extend evaluation to LLMs and situated contexts (Tan et al., 2023; Chu et al., 2024; Wang and Zhao, 2024; Wei et al., 2025; Chen et al., 2021), they still do not explicitly evaluate tool-augmented, iterative temporal reasoning

To address this gap, we propose Time Puzzles, a family of simple, puzzle-like constraint-based date inference tasks designed to evaluate *precise, iterative* temporal reasoning with tools. As illustrated in Figure 1, each puzzle presents natural-language temporal constraints that combine factual temporal anchors with (cross-cultural) calendar-based relations, and asks the model to infer the date(s) that jointly satisfy all constraints. Solving a puzzle requires *iteratively proposing, refining, and verifying candidate dates* using external tools (e.g., calendars or web search), emphasizing constraint-based reasoning rather than rote memorization or single-shot recall. This formulation reflects real-world settings such as scheduling, planning, and historical analysis, where systems must integrate factual information with calendar structure to produce temporally consistent decisions.

*Equal contributions. ZW conceived the research idea. ZD designed and implemented the algorithm for dataset generation. ZW designed and executed the LLM experiments. Both authors contributed significantly to drafting and revising the manuscript. ZW revised the second version of this draft.

Findings of the Association for Computational Linguistics: IJCNLP-AACL 2026, pages 683–694
November 6-10, 2026 ©2026 Association for Computational Linguistics

We evaluate 13 diverse LLMs on Time Puzzles under both tool-less and tool-augmented settings. Overall performance remains limited: even the best model (GPT-5) reaches only 55.3% accuracy without tools, despite the use of easily searchable factual anchors. While web search consistently improves results, all models perform better when the same puzzles are rewritten to replace factual anchors with explicit dates. This gap highlights a key weakness in integrating factual lookup with constraint-based, multi-step temporal reasoning.

2 Time Puzzles

This section defines the Time Puzzles task and describes its algorithmic generation process.

2.1 Task Formulation

Time Puzzles is a constraint-based date inference task: given a set of temporal constraints, the goal is to identify the date(s) that jointly satisfy all constraints. Formally, let $\mathcal{D}$ denote the set of all possible Gregorian dates, and let $\mathcal{C}$ be an oracle function that maps a natural-language temporal constraint t to the set of dates for which t holds, i.e., $\mathcal{C}(t) \subseteq \mathcal{D}$. As illustrated in Figure 1, each puzzle consists of N natural-language constraints $F = \{t_1, t_2, \ldots, t_N\}$. The answer set $\mathcal{A}$ contains all dates that satisfy every constraint:

$$\mathcal{A} = \bigcap_{i=1}^{N} \mathcal{C}(t_i) \quad (1)$$

We restrict our study to puzzles with non-empty answer sets ($|\mathcal{A}| \geq 1$) to focus on LLMs' ability to identify valid solution dates via iterative temporal reasoning, and cap $|\mathcal{A}|$ at 6 for exploration.

2.2 Data Generation

Input Constraints We create a simple yet diverse taxonomy of temporal constraint types (Table 3 in Appendix A.2). The taxonomy spans both factual anchors (e.g., historical events, presidents in office, zodiac years) and calendar-structural constraints (e.g., months, seasons, weekdays, or day-of-month relations), each restricting candidate dates at the year, month, or day level.

Because Time Puzzles is designed to probe tool-augmented iterative reasoning, we append to each puzzle a randomly sampled, trivial historical event from a curated list of 50 that *potentially* require web search. All events occur no later than 2023 and are *easily searchable*, as manually verified during curation (Appendix A.1). We leave harder-to-find facts for future stress-testing.

This paper considers only English as a proof of concept, but Time Puzzles can be readily adapted to other languages. Figure 1 shows an example puzzle with five constraints.

Puzzle Generation Puzzles are generated algorithmically through randomized constraint sampling. Each puzzle combines one of the 50 real-world historical events we manually curated with additional temporal constraints from the taxonomy (e.g., calendar relations). We retain combinations that yield puzzles with a desired number of valid solution dates. We open-source our code and generated datasets at `https://github.com/jaaack-wang/Time-Puzzles`.

Resulting Datasets We construct two conceptually equivalent datasets for controlled evaluation. The default dataset uses implicit constraints (e.g., an event) to target tool-augmented iterative temporal reasoning. The paired dataset rewrites each implicit constraint by spelling out its relevant dates explicitly, thereby removing the need of factual lookup. Each dataset contains 600 puzzles, with solution set sizes evenly distributed from 1 to 6.

We select 100 puzzles per solution size due to cost constraints, as unrestricted web search is prohibitively expensive, given our limited budget. In addition, experiments on a separate set of 600 puzzles yield similar results, supporting the suitability of the current dataset size (see Section 4).

Quality Control Because all constraints are independently verifiable, the resulting puzzles are both scalable and high quality. We randomly inspected 30 puzzles, where one author independently attempted to solve them using a web browser and verified the correctness of the generated puzzles. This manual inspection also yielded an estimated human baseline accuracy of 80%.

3 Experiments

This section first outlines the overall experimental setup and then reports results on Time Puzzles under both tool-free and tool-augmented settings.

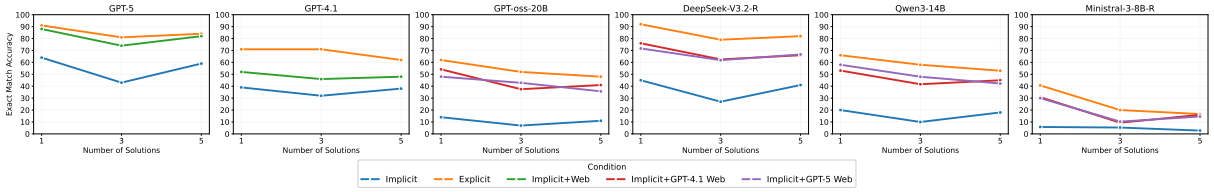


Figure 2: Average exact match accuracy across solution counts, with and without web search (only on solution counts 1, 3, 5). GPT-4.1/5 run live web search; for open-weight models we re-use the same **cited** GPT web results.

Model	EM (%)	JI (%)	F1 (%)	# Tks
GPT-5	55.33	58.62	59.52	4126
GPT-5-nano	15.83	19.37	20.31	5224
GPT-4.1	35.17	39.13	40.14	615
GPT-4.1-nano	9.33	13.53	14.84	1098
GPT-oss-20B	10.83	13.51	14.32	5704
DeepSeek-V3.2-R	36.67	39.33	40.08	7862
DeepSeek-V3.2	30.50	32.89	33.51	737
Qwen3-4B-Inst	3.00	3.89	4.14	2408
Qwen3-4B-Think	5.67	6.61	6.89	9701
Qwen3-8B	10.83	13.45	14.26	7341
Qwen3-14B	13.17	16.32	17.36	4733
Ministral-3-8B-Inst	3.50	5.38	6.04	1797
Ministral-3-8B-R	4.27	4.81	5.04	2058

Table 1: Exact-match accuracy (EM), Jaccard index (JI), F1 score (F1), and output tokens (Tks), averaged over six solution counts, for the 600 generated puzzles with implicit (default) constraints in the tool-less setting.

3.1 Experimental Setup

Models We evaluate 13 diverse LLMs from four model families (Table 1) released in 2025. These include four proprietary GPT models (GPT-5 and GPT-4.1 series) (OpenAI, 2025b) and nine open-weight models: GPT-oss-20B (OpenAI et al., 2025) plus instruction/reasoning variants from DeepSeek-V3.2 (DeepSeek-AI et al., 2025), Qwen3 (Yang et al., 2025), and Ministral-3-8B (Mistral AI, 2025). See Table 4 in Appendix B for further details.

Prompting All models are prompted using zero-shot chain-of-thought (CoT) prompting (Kojima et al., 2022) to encourage step-by-step reasoning. The prompt template is provided in Appendix C.

Metrics Let Y_i denote the gold solution set and $\hat{Y}_i$ the predicted set for instance i. We compute exact match (EM), F1 (F1), and Jaccard index (JI):

$$EM_i = \mathbb{I}[Y_i = \hat{Y}_i]. \tag{2}$$

$$P_i = \frac{|Y_i \cap \hat{Y}_i|}{|\hat{Y}_i|}, \; R_i = \frac{|Y_i \cap \hat{Y}_i|}{|Y_i|}, \; F1_i = \frac{2P_iR_i}{P_i+R_i}. \tag{3}$$

$$JI_i = \begin{cases} 1 & Y_i = \hat{Y}_i = \emptyset \\ \frac{|Y_i \cap \hat{Y}_i|}{|Y_i \cup \hat{Y}_i|} & \text{otherwise.} \end{cases} \tag{4}$$

Given our emphasis on *precise* iterative reasoning, we treat EM as the primary metric.

Tools We consider the following tool-augmented settings: web search, Code Interpreter (+CI), and their combinations. Due to cost constraints, these experiments are conducted on puzzles with solution counts 1, 3, and 5 and with selected models.

More specifically, for web search, we test six best-performing LLMs across different model families (GPT-5, GPT-4.1, GPT-oss-20B, DeepSeek-V3.2-R, Qwen3-14B, and Ministral-3-8B-R), whereas for +CI, we only evaluate GPT-5 and GPT-4.1 both with and without web search. We use off-the-shelf tools via OpenAI Response API and we include GPT-5 and GPT-4.1 across all tool-augmented settings to represent tool-using capablities by a reasoning model (GPT-5) as well as an instruction model (GPT-4.1). Note that GPT-5 and GPT-4.1 are run with live web search and their retrieved results are reused as fixed context for the open-weight models.

3.2 Results

Despite its simplicity, Time Puzzles is challenging for tool-less LLMs. Our dataset is a *proof of concept* built from easy-to-search facts that predates each model's training cutoff or release date. Yet even GPT-5 achieves only 55.3% EM on the dataset, with all other models below 37% EM (Table 1). JI and F1 yield slightly higher scores, but the overall performance patterns remain consistent, with a Spearman correlation of $r = 0.93$. Moreover, the three metrics agree in 88.6% of cases, indicating that LLMs typically either solve a puzzle completely or fail to solve it at all.

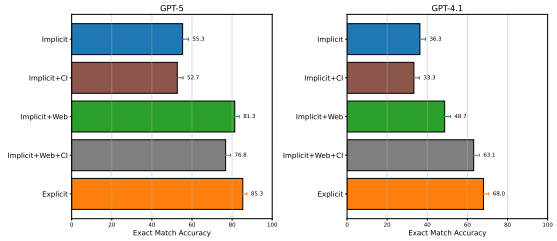


Figure 3: Average EM accuracy (%) for GPT-5 (left) and GPT-4.1 (right) under different conditions across three solution counts (1, 3, and 5). Numbers annotated on bars are mean values, beside error bars. `+Web` enables web search. `+CI` enables Code Interpreter.

Type	Setting	G4.1-Nano	G5-Nano	G4.1	G5
Implicit	Default	9.3	15.8	35.2	55.3
	+Reason Inst	8.8	14.2	25.3	47.3
	New Runs	11.0	16.8	29.8	50.1
Explicit	Default	41.8	60.2	63.0	84.7
	+Reason Inst	35.5	55.5	53.5	77.7
	New Runs	42.7	60.2	63.7	80.5

Table 2: EM accuracy (%) averaged over six solution counts under different settings for both explicit and implicit constraint across the four GPT models. Default setting is based on Section 3. "+Reason Inst" enhances the default setting with detailed step-by-step reasoning instruction. "New Runs" reruns the default setting with completely new puzzles generated from scratch.

Tool use, particularly web search, can greatly improve model performance, but a clear reasoning gap remain. Figure 2 shows that web search consistently improves performance across different solution counts, but models nearly always perform the best when the implicit constraints in Time Puzzles are rewritten with explicit dates to remove the need for factual lookup. Similarly, Figure 3 shows that enabling Code Interpreter (+CI) does not close the gap either: +CI degrades GPT-5 performance on implicit constraints, both with and without web search. While GPT-4.1 benefits from +CI when combined with web search, the performance is below that of explicit constraints.

4 Analysis

Section 3 shows that Time Puzzles provides a challenging and cost-effective benchmark for iterative temporal reasoning. This section further demonstrates that as a benchmark, Time Puzzles is both discriminative and reliable.

Time Puzzles effectively distinguishes nuanced iterative temporal reasoning capabilities. Table 1 clearly shows that larger models consistently outperform their smaller counterparts (e.g., GPT-X vs. GPT-X-nano; Qwen3 variants), and reasoning models uniformly surpass instruction-tuned ones (e.g., DeepSeek/Ministral pairs). However, scale alone is insufficient: despite its smaller size, Qwen3-4B-Think substantially outperforms Ministral-3-8B-R, and GPT-oss-20B similarly surpasses both while using only 3.6B active parameters. Moreover, for reasoning models, higher performance typically uses less output tokens, suggesting that success on Time Puzzles hinges on sustained, structured reasoning rather than exhaustive generation (e.g., Qwen3 models in Table 1).

Our findings are robust to prompt variations and puzzle sets. Table 2 shows that inserting a detailed step-by-step reasoning instruction into the default prompt (see Appendix C) decreases performance, particularly for larger models. Furthermore, repeating the experiments from Section 3 on 600 newly generated puzzles yields negligibly different exact-match (EM) scores, validating the suitability of our evaluation dataset size.

5 Related Work

Temporal reasoning has long been studied in NLP, including event ordering, implicit event inference, and temporal commonsense reasoning (Ning et al., 2020; Zhou et al., 2021, 2019; Qin et al., 2021). More recent work focuses on diagnosing temporal reasoning in LLMs through broader benchmarks and systematic analyses across diverse task formats (Tan et al., 2023; Chu et al., 2024; Wang and Zhao, 2024; Wei et al., 2025), as well as targeted probes of specific temporal phenomena and synthetic constructions (Islakoglu and Kalo, 2025; Wei et al., 2023; Fatemi et al., 2025; Bhatia et al., 2025).

A complementary line of work studies temporal reasoning in situated and dynamic contexts (Chen et al., 2021; Zhang and Choi, 2021; Kasai et al., 2023). Other studies explore more agentic settings that incorporate memory, external computation, or multimodal inputs (Ge et al., 2025; Saxena et al., 2025). Overall, existing benchmarks largely evaluate temporal reasoning in static or retrieval-oriented settings. In contrast, Time Puzzles focus on constraint-based date inference that benefits from explicit tool use and requires iterative

reasoning, providing a complementary evaluation of tool-augmented, agentic temporal reasoning.

6 Conclusion

We propose Time Puzzles, a constraint-based date inference task that targets iterative, tool-augmented temporal reasoning. Although puzzles are synthetically generated and easy to verify, they are challenging for tool-less LLMs. They also expose persistent failures of current instruction- and reasoning-tuned LLMs to reliably resolve implicit temporal constraints, even with tool access. Overall, Time Puzzles offers a simple, cost-effective, and discriminative diagnostic for tool-augmented iterative temporal reasoning and can be systematically extended to support more challenging evaluations or additional languages (e.g., via translation).

Acknowledgement

Zhengxiang Wang is grateful for support from the Institute for Advanced Computational Science (IACS) at Stony Brook University. We also thank the three anonymous reviewers for their valuable feedback.

Limitations

Synthetic Dataset Construction Time Puzzles is enabled by a controllable generation algorithm, but the resulting constraints are not guaranteed to match real-world temporal reasoning in full complexity. In particular, the naturalness of the constraints can vary, and template-based composition may miss ambiguity, conflicting evidence, or underspecified conditions that arise in practice.

Tool and Model Scope Our experiments do not exhaust the space of models, tools, or agentic workflows. We evaluate off-the-shelf LLMs and off-the-shelf tools, and we do not attempt to build sophisticated tool-use policies to solve Time Puzzles, given the scope of the study. As a result, our results should be interpreted as a diagnostic of current out-of-the-box capabilities rather than an upper bound.

Solution-Less Puzzles We restrict evaluation to puzzles with at least one valid solution in order to focus on LLMs' ability to identify correct dates via iterative temporal reasoning. Although our generation algorithm can produce puzzles with zero solutions, such instances often contain trivial self-contradictions (e.g., incompatible year, month, or day constraints) that require little reasoning and are easily solved by smaller models. Designing non-trivial solution-less puzzles that demand substantive iterative reasoning is left to future work.

Language Diversity Our work currently only considers English language, due to budget constaints. Future studies can extend our study by adapting our evaluation framework to other languages. This can be easily done by, for example, converting the puzzle template into other languages.

Ethical Considerations

Data and privacy. We do not collect human subjects data. Time Puzzles instances are synthetically generated, and our factual anchors are based on public historical information, with no personally identifiable information.

References

Gagan Bhatia, Ming Ze Tang, Cristina Mahanta, and Madiha Kazi. 2025. DateLogicQA: Benchmarking temporal biases in large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 4: Student Research Workshop)*, pages 321–332, Albuquerque, USA. Association for Computational Linguistics.

Wenhu Chen, Xinyi Wang, William Yang Wang, and William Yang Wang. 2021. A dataset for answering time-sensitive questions. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1.

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)*, pages 1204–1228, Bangkok, Thailand. Association for Computational Linguistics.

DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, and 245 others. 2025. Deepseek-v3.2: Pushing the frontier of open large language models. *Preprint*, arXiv:2512.02556.

Bahare Fatemi, Mehran Kazemi, Anton Tsitsulin, Karishma Malkan, Jinyeong Yim, John Palowitch, Sungyong Seo, Jonathan Halcrow, and Bryan Perozzi. 2025. Test of time: A benchmark for evaluating LLMs on temporal reasoning. In *The Thirteenth International Conference on Learning Representations*.

Yubin Ge, Salvatore Romeo, Jason Cai, Raphael Shu, Yassine Benajiba, Monica Sunkara, and Yi Zhang. 2025. TReMu: Towards neuro-symbolic temporal reasoning for LLM-agents with memory in multi-session dialogues. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 18974–18988, Vienna, Austria. Association for Computational Linguistics.

Duygu Sezen Islakoglu and Jan-Christoph Kalo. 2025. ChronoSense: Exploring temporal understanding in large language models with time intervals of events. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 590–602, Vienna, Austria. Association for Computational Linguistics.

Jungo Kasai, Keisuke Sakaguchi, yoichi takahashi, Ronan Le Bras, Akari Asai, Xinyan Velocity Yu, Dragomir Radev, Noah A. Smith, Yejin Choi, and Kentaro Inui. 2023. Realtime QA: What's the answer right now? In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Mistral AI. 2025. Introducing mistral 3. https://mistral.ai/news/mistral-3. Accessed: 2025-12-27.

Qiang Ning, Hao Wu, Rujun Han, Nanyun Peng, Matt Gardner, and Dan Roth. 2020. TORQUE: A reading comprehension dataset of temporal ordering questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1158–1172, Online. Association for Computational Linguistics.

OpenAI. 2025a. Gpt-5. https://openai.com. Large language model.

OpenAI. 2025b. Introducing gpt-4.1 in the api. https://openai.com/index/gpt-4-1/. Accessed: 2025-12-26.

OpenAI, Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett, Eugene Brevdo, Greg Brockman, Sebastien Bubeck, Che Chang, and 107 others. 2025. gpt-oss-120b & gpt-oss-20b model card. *Preprint*, arXiv:2508.10925.

OpenRouter.ai. 2025. State of AI: An empirical 100 trillion token study with openrouter. https://openrouter.ai/state-of-ai. Accessed: 2025-12-26.

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, Online. Association for Computational Linguistics.

Rohit Saxena, Aryo Pradipta Gema, and Pasquale Minervini. 2025. Lost in time: Clock and calendar understanding challenges in multimodal llms. *Preprint*, arXiv:2502.05092.

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, Toronto, Canada. Association for Computational Linguistics.

Yuqing Wang and Yun Zhao. 2024. TRAM: Benchmarking temporal reasoning for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 6389–6415, Bangkok, Thailand. Association for Computational Linguistics.

Shaohang Wei, Wei Li, Feifan Song, Wen Luo, Tianyi Zhuang, Haochen Tan, Zhijiang Guo, and Houfeng Wang. 2025. Time: A multi-level benchmark for temporal reasoning of LLMs in real-world scenarios. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

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, Singapore. Association for Computational Linguistics.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao

Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Michael Zhang and Eunsol Choi. 2021. SituatedQA: Incorporating extra-linguistic contexts into QA. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7371–7387, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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, Hong Kong, China. Association for Computational Linguistics.

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, Online. Association for Computational Linguistics.

A Data Generation

A.1 Factual Anchor Collection

Collection Procedure We manually curate 50 easy-to-search factual instances based on TEMPREASON (Tan et al., 2023), a time-sensitive question answering dataset designed to evaluate temporal reasoning over time–event and event–event relations. Our collection procedure is as follows:

- We randomly sample 400 questions involving time–event and event–event relations from TEMPREASON. Each instance consists of a factual question (e.g., "Which position did Siim Kallas hold in September 1999?") and its corresponding answer (e.g., "Minister of Finance").
- We prompt `gpt-oss-20b` to answer the sampled questions and find that it could barely answer these questions.
- For each retained instance, we reframe the question–answer pair into a declarative statement describing an event, with the temporal information removed. We then use web search to recover the corresponding time point or time range during which the event occurred. When necessary, we further reframe the instance to query a closely related event with an unambiguous temporal answer. We stop once 50 valid instances are collected. **All temporal answers are obtained from top-ranked Google search results and manually verified by the authors.**
- We use browser-based ChatGPT-5.2-Auto and Gemini 3 Pro with web search enabled to independently fact-check the complete fact table (provided via screenshots). Minor typographical errors are corrected, and one ambiguous instance is replaced. After revision, both models identify no remaining issues.

Fact-Checking Prompt Instruction See below.

```
You are given a table containing 50
historical events. For each event, perform a
 web search to verify whether the provided
time information is accurate.

Column meanings:

- start: The year (or date) when the event
first began.

- end: The year (or date) when the event
ended.

- multi_year_spans: The range of years
during which the event took place.

Task instructions:

- Verify whether the provided time
information for each event is correct or
incorrect based on reliable sources.

- The time may be given at different levels
of precision (e.g., year-only vs. full date).


- Judge correctness only at the precision
provided.

- Do not mark a time as incorrect solely
because it omits a more specific month or
day.

Output requirements (one entry per event):
For each event, report:

- The row index

- The event name

- Whether the provided time is correct or
incorrect

- A brief explanation supporting your
judgment
```

Including Well-Known Facts To balance the corpus, we additionally include 20 widely known factual instances. These cover the presidencies of recent U.S. presidents (since 1945) and the lifespans of well-known public figures (e.g., Kobe Bryant, Albert Einstein).

A.2 Algorithm Implementation Details

We generate each puzzle by sampling a hidden *seed date* and selecting N natural-language constraints whose conjunction yields a user-defined number of solutions. To ensure diversity and computational efficiency, our pipeline utilizes a comprehensive taxonomy of temporal facts and sorts constraints by *Information Gain* (IG) during verification.

Fact Taxonomy and Constraint Levels To ensure diversity, we define a set of fact categories $\mathcal{K}$. Each category represents a specific type of temporal knowledge (e.g., calendar structure, historical events, astronomical cycles). Furthermore, we assign a *Constraint Level* $L(t) \in \{\textsc{Year}, \textsc{Month}, \textsc{Day}\}$ to each fact type, indicat-

ing the granularity of the information provided. Table 3 summarizes the fact categories used in Time Puzzles.

Entropy and Information Gain To generate efficient solvers and measure the utility of each clue, we define the IG for a fact t. Let the entropy of the date space given a set of constraints be the logarithm of the size of the valid date set. The information gain of a fact t is the reduction in entropy it provides:

$$IG(t) = \log_2(|\mathcal{D}|) - \log_2(|\mathcal{C}(t)|) \quad (5)$$

Facts with higher IG narrow the search space more aggressively. For example, an exact date has maximal IG, whereas "It is a Monday" has low IG.

Algorithm Details We provide the specific procedural logic for generating diverse fact sets and solving the puzzles, as referenced in the main methodology. The fact generation process, detailed in Algorithm 1, ensures diversity by selecting facts from different categories and balanced constraint levels. The solution verification process, detailed in Algorithm 2, optimizes performance by first filtering with facts with high IG.

Algorithm 1: GENERATEFACT Procedure

Input: Anchor t_{kb}, Seed d_{seed}, Count N, Universal Date Set $\mathcal{D}$
Output: Set of facts F

$F \leftarrow \{t_{kb}\}$;
$UsedCats \leftarrow \{Cat(t_{kb})\}$;
// Iteratively generate N-1 distinct facts
for $i \leftarrow 1$ **to** $N-1$ **do**
  // Ensure balanced levels
  $TargetLevel \leftarrow$ DetermineLevel(i, N);
  $CandidateCats \leftarrow \{k \in \mathcal{K} \mid k \notin UsedCats, Level(k) == TargetLevel\}$;
  $k_{new} \leftarrow$ RandomSelect$(CandidateCats)$;
  $t_{new} \leftarrow$ InstantiateFact(k_{new}, d_{seed});
  $F \leftarrow F \cup \{t_{new}\}$;
  $UsedCats \leftarrow UsedCats \cup \{k_{new}\}$;
end
return F;

Algorithm 2: SOLVEPUZZLE Procedure

Input: Set of facts F, Universal Date Set $\mathcal{D}$
Output: Solution set $\mathcal{A}$
// Sort facts to optimize intersection
Compute $IG(t)$ for all $t \in F$;
$F_{sorted} \leftarrow$ SortDescending$(F, \text{key} = IG)$;
$\mathcal{A} \leftarrow \mathcal{D}$;
for $t \in F_{sorted}$ **do**
  $\mathcal{A} \leftarrow \mathcal{A} \cap \mathcal{C}(t)$;
end
return $\mathcal{A}$;

B Models

Inference Details We evaluate four OpenAI GPT-family models via the OpenAI API (GPT-5, GPT-5-nano, GPT-4.1, and GPT-4.1-nano), using the dated snapshot identifiers shown in Table 4. We also evaluate two DeepSeek-V3.2 endpoints via the DeepSeek API (`deepseek-chat` and `deepseek-reasoner`; accessed in December 2025). For open-weight models, we run local inference with vLLM (Kwon et al., 2023). We use default generation configuration for all models.

Model Specification Table 4 lists the exact model variants (API snapshot IDs or Hugging Face repo IDs), parameter sizes, release dates, and (when disclosed) knowledge cutoffs.

Web Search Implementation We enable web search for GPT-4.1 and GPT-5 via the official Responses API by providing the web search tool at inference time and setting tool choice as *required*. From each model's output JSON, we extract the URLs cited in its response and scrape the corresponding webpage contents to construct a fixed web-search context for all open-weight LLMs. This context is prepended to the prompt in the form: "`Here is some web search context that may help you: {scraped contents from cited URLs}.`"

Code Interpreter Implementation We enable Code Interpreter for GPT-4.1 and GPT-5 via the official Responses API, with tool choice set as *required*.

Web Search + Code Interpreter Implementation Providing Web Search and Code Interpreter simultaneously does not guarantee that both tools are invoked during generation, even when tool choice is set as *required*. To control costs and ensure consistent retrieval, we therefore reuse the web-search context collected from the Web Search-only setting and prepend it to the prompt for GPT models, while enabling Code Interpreter with tool choice set as *required*.

C Prompts

C.1 Default Prompt

We use the following prompt template in the main experiments in Section 3. The placeholder

Fact Name	Description	Level
ExplicitYearFact	Explicitly states the year (e.g., "The year is 2025").	YEAR
DecadeFact	Specifies the decade (e.g., "The 1990s").	YEAR
LeapYearFact	States that the year is a leap year.	YEAR
ChineseZodiacFact	Specifies the Chinese Zodiac animal.	YEAR
PersonAliveFact	States a famous person (e.g., Steve Jobs) was alive.	YEAR
USPresidentFact	States a specific US President was in office.	YEAR
EventFact	States if it is an Olympic or World Cup year.	YEAR
MonthFact	Specifies the month of the year.	MONTH
SeasonFact	Specifies the season (e.g., Winter) based on month.	MONTH
LunarMonthFact	Specifies the month in the Chinese lunar calendar.	MONTH
WeekdayInMonthFact	Specifies the Nth occurrence of a weekday (e.g., "2nd Friday").	DAY
DayOfMonthFact	Specifies the exact day or if it is the first/last day.	DAY
WeekdayFact	Specifies the day of the week (e.g., "It is Monday").	DAY
MultiWeekdayFact	Specifies a set of possible weekdays (e.g., ůůMon or Tue").	DAY
DayOfMonthRangeFact	Specifies if the day is before/after a specific day.	DAY
KnowledgeBaseEventFact	Relates the date to a historical event (same day/-month/year), such as the 50 facts collected in Appendix A.1.	VARIOUS

Table 3: Taxonomy of temporal facts used in generation, categorized by constraint granularity level.

{constraints} is replaced with the instance-specific constraint set at inference time.

```
From the time-related constraints below,
determine all valid date(s) (if any) that
satisfy them. Depending on the conditions,
the result may be zero, one, or multiple
dates. Unless otherwise stated, interpret
all constraints using the Gregorian calendar.


Note: Seasons are defined as:
- Winter: December, January, February
- Spring: March, April, May
- Summer: June, July, August
- Autumn: September, October, November

The constraints are as follows:

{constraints}

Carefully review the constraints and reason
step-by-step to identify all valid date(s).
After thorough consideration, end your
response on a new line with "MY ANSWER: "
followed by the valid date(s) in the format
"YYYY-MM-DD". If there are multiple valid
dates, list them separated by commas. If no
valid date exists, respond with "MY ANSWER:
None".
```

C.2 Prompt with Step-by-step Reasoning Instruction

The following prompt is used in the follow-up experiment in Section 4, which builds on the default prompt by inserting detailed step-by-step reasoning instruction.

```
From the time-related constraints below,
determine all valid date(s) (if any) that
satisfy them.
Depending on the conditions, the result may
be zero, one, or multiple dates.
Unless otherwise stated, interpret all
constraints using the Gregorian calendar.

Carefully follow these step-by-step
reasoning instructions:

1) Read and restate the goal
   - You must find ALL Gregorian calendar
date(s) that satisfy EVERY constraint.
   - The answer can be none, one, or many
dates.

2) Normalize definitions you must use
   - Seasons:
     * Winter = Dec, Jan, Feb
     * Spring = Mar, Apr, May
     * Summer = Jun, Jul, Aug
     * Autumn = Sep, Oct, Nov
   - If constraints mention weekday, leap
year, end of month, etc., use standard
Gregorian rules.
```

Model	Variant used (snapshot / HF repo)	Params	Release date	Knowledge cutoff
GPT-5 (OpenAI, 2025a)	`gpt-5-2025-08-07`	Not disclosed	2025-08-07	2024-09-30
GPT-5-nano (OpenAI, 2025a)	`gpt-5-nano-2025-08-07`	Not disclosed	2025-08-07	2024-05-31
GPT-4.1 (OpenAI, 2025b)	`gpt-4.1-2025-04-14`	Not disclosed	2025-04-14	2024-06-01
GPT-4.1-nano (OpenAI, 2025b)	`gpt-4.1-nano-2025-04-14`	Not disclosed	2025-04-14	2024-06-01
GPT-oss-20B (OpenAI et al., 2025)	`openai/gpt-oss-20b`	20.9B (3.6B active)	2025-08-05	2024-06
DeepSeek-V3.2-R (DeepSeek-AI et al., 2025)	`deepseek-reasoner`	685B	2025-12-01	Not disclosed
DeepSeek-V3.2 (DeepSeek-AI et al., 2025)	`deepseek-chat`	685B	2025-12-01	Not disclosed
Qwen3-4B-Inst (Yang et al., 2025)	`Qwen/Qwen3-4B-Instruct-2507`	4B	2025-08-06	Not disclosed
Qwen3-4B-Think (Yang et al., 2025)	`Qwen/Qwen3-4B-Thinking-2507`	4B	2025-08-06	Not disclosed
Qwen3-8B (Yang et al., 2025)	`Qwen/Qwen3-8B`	8B	2025-04-29	Not disclosed
Qwen3-14B (Yang et al., 2025)	`Qwen/Qwen3-14B`	14B	2025-04-29	Not disclosed
Ministral-3-8B-Inst (Mistral AI, 2025)	`mistralai/Ministral-3-8B-Instruct-2512`	8B	2025-12-02	Not disclosed
Ministral-3-8B-R (Mistral AI, 2025)	`mistralai/Ministral-3-8B-Reasoning-2512`	8B	2025-12-02	Not disclosed

Table 4: Model variants and specifications used in our experiments. We identify OpenAI models by their dated snapshot IDs and open-weight models by their Hugging Face repository IDs. Release dates reflect snapshot timestamps or official provider documentation. Note that precise release and knowledge cutoff dates may vary slightly across sources.

```
3) Extract constraints into a checklist
   - Parse the text and list each constraint
 as a separate item.
   - If any constraint contains an OR, split
 into separate branches.
   - If constraints contain ranges (e.g., "
between 2010 and 2015"), record inclusivity/
exclusivity exactly as stated.

4) Determine the search space (candidate
window)
   - Identify the tightest restriction on:
     * Year(s) (specific year, range,
century, decade)
     * Month(s) (specific month(s), season,
quarter)
     * Day(s) (day number, end of month,
first Monday, etc.)
   - Use these to form an initial set of
candidate dates or a candidate-generating
rule.

5) Convert each constraint into a filter
function
   For each constraint, translate it into a
precise test such as:
   - Year filter (e.g., year = 2024; year in
 [1990, 1999])
   - Month/season filter (e.g., month in
{{6,7,8}})
   - Day-of-month filter (e.g., day = 1; day
 >= 28)
   - Weekday filter (e.g., weekday = Monday)
   - Relative-date relations (e.g., "the day
 after X", "two weeks before Y")
   - Calendar properties:
     * Leap year rule: leap if divisible by
4, except centuries not divisible by 400.
     * Month lengths: 30/31 days; February
28 or 29 depending on leap year.

6) Generate candidates in the most
restrictive way
   - Prefer generating candidates from the
most specific constraints first:
     * If a specific date is implied,
compute it directly.
     * If a specific year + season is given,
 iterate only those months.
     * If a weekday pattern exists ("first
Monday of May"), generate those directly
instead of brute-forcing all days.

7) Apply constraints systematically
   - For each candidate date, check
constraints one by one.
   - Eliminate candidates immediately when
one constraint fails.
   - Keep a running list of remaining
candidates.

8) Handle OR and branching logic correctly
   - If constraints are like: (A OR B) AND C:

     * Compute candidates satisfying A and C
     * Compute candidates satisfying B and C
     * Take the union of those results (
deduplicate).

9) Check for hidden contradictions early
   - Before enumerating too much, look for
impossibilities, e.g.:
     * Month is February but day is 30
     * Season says Winter but month says
July
     * Date required to be both Monday and
Tuesday
     * Before 2000 AND after 2010
   - If a contradiction is found, conclude
no solution.

10) Ensure completeness (don't miss valid
dates)
   - Confirm you explored all years/months
allowed by the constraints.
   - If you used branching, ensure all
branches were evaluated.

11) Sort and format final answers
```

```
   - Sort valid dates chronologically.
   - Output dates in YYYY-MM-DD format.
   - If multiple, separate by commas and a
space.

The constraints are as follows:

{constraints}

After thorough consideration, end your
response on a new line with "MY ANSWER: "
followed by the valid date(s) in the format
"YYYY-MM-DD".
If there are multiple valid dates, list them
 separated by commas.
If no valid date exists, respond with "MY
ANSWER: None".
```