Measuring Iterative Temporal Reasoning with Time Puzzles

Anonymous ACL submission

Abstract

We introduce Time Puzzles, a constraint-based date inference task for evaluating iterative temporal reasoning. Each puzzle combines factual temporal anchors with (cross-cultural) calendar relations, admits one or multiple valid solution dates, and is algorithmically generated for controlled, dynamic, and continual evaluation. Across 13 diverse LLMs, Time Puzzles well distinguishes their iterative temporal reasoning capabilities and remains challenging without tools: GPT-5 reaches only 49.3% accuracy and all other models stay below 31%, despite the dataset's simplicity. Web search consistently yields substantial gains and using code interpreter shows mixed effects, but all models perform much better when constraints are rewritten with explicit dates, revealing a gap in reliable tool use. Overall, Time Puzzles presents a simple, cost-effective diagnostic for tool-augmented iterative temporal reasoning.

1 Introduction

Time is a fundamental attribute of human societies, and temporal reasoning is a core component of human intelligence. In practice, humans rarely reason about time in isolation or by rote memorization; instead, they routinely rely on external tools such as clocks, calendars, written records, and online resources to support temporal inference and verification. With the growing deployment of tool-using, agentic LLMs across consumer-facing, research, and industrial settings (OpenRouter.ai, 2025), it is both timely and important to evaluate iterative temporal reasoning in tool-augmented contexts.

Despite extensive study, existing temporal reasoning benchmarks remain mismatched with how temporal reasoning is performed in practice. Prior work largely evaluates temporal understanding in 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.

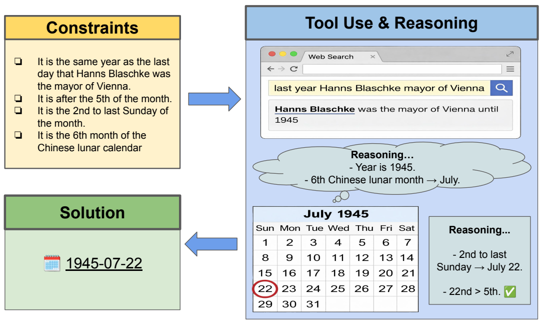

Figure 1: Time Puzzles: a simple date inference task requiring iterative tool-aided temporal reasoning.

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.

Our main contributions are threefold:

1. We introduce Time Puzzles, a novel and dynamic benchmark for evaluating iterative temporal reasoning with tools. The benchmark is enabled by a controllable generation

algorithm that supports continual evaluation beyond static test sets prone to memorization.

2. We conduct a comprehensive evaluation across 13 diverse LLMs. Although tool access improves accuracy, substantial gaps remain in producing fully consistent solutions, highlighting the need for more reliable tool use. We show that Time Puzzles serves as a practical diagnostic for progress in tool-augmented iterative temporal reasoning.

3. For reproducibility, we release the code and datasets for generating challenging constraint-based time puzzles.[1]

2 Time Puzzles

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

2.1 Task Formulation

Let $\mathcal{D}$ denote the set of all possible Gregorian dates, and let $\mathcal{C}$ map a natural language fact 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 facts $F = \{t_1, t_2, \ldots, t_N\}$, with an answer set $\mathcal{A}$ that satisfies all constraints jointly:

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

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

Algorithm We generate Time Puzzles instances using the dynamic procedure in Algorithm 1. The algorithm takes as input a set of pre-configured temporal fact templates and a target solution size M, and returns a puzzle instance $(F, \mathcal{A})$ with $|\mathcal{A}| = M$.

Generation proceeds as a randomized search-and-find process. We first anchor the puzzle with a real-world event description t_{kb} (often requiring web search) and sample a *seed date* d_{seed} that is temporally consistent with t_{kb}. We then randomly sample 3-5 more temporal constraints (e.g., calendar relations) until a solver yields an answer set of size M. To improve data generation efficiency, the solver prunes the search space using an entropy-based approach (see Appendix A.2 for details). For our datasets detailed below, it takes around 2 seconds to generate each puzzle in a single core CPU.

[1] Will be released upon acceptance, for anonymity.

Algorithm 1: TimePuzzles Data Generation Process

```
Input: Target facts N, Target solution size M,
       Universal Date Set D
Output: A puzzle instance (F, A)
while True do
    // Step 1: Randomly pick a seed date
    t_kb ← SelectRandomKnowledgeBasedEvent();
    d_seed ← RandomDate(C(t_kb));
    // Step 2: Generate Facts
    // See Appendix A.2 for detail
    F ← GenerateFact(t_kb, d_seed, N, D);
    // Step 3: Solve and Validate
    // See Appendix A.2 for detail
    A ← SolvePuzzle(F, D);
    // Step 4: Output condition
    if |A| == M then
        return (F, A);
    end
end
```

Temporal Facts To ensure diversity and realism, we create a diverse taxonomy of temporal facts (Table 2 in Appendix A.2) that supports constraints at three granularities: year, month, and day. Our facts span lesser-known historical events, well-known anchors, and calendar relations (e.g., day-of-week and non-Gregorian calendars), among others.

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.

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, isolating pure iterative temporal reasoning. Each dataset contains 600 puzzles, with solution set sizes evenly distributed from 1 to 6.

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.

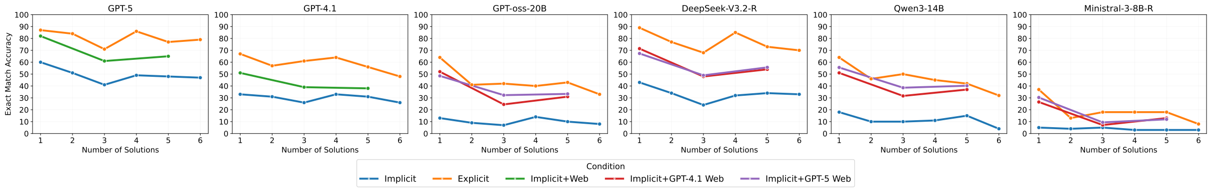

Figure 2: Average exact match accuracy across solution counts, with and without web search (only on solution counts 1, 3, 5). GPT-4.1/5 run live web search; for open-weight models we re-use the same **cited** GPT web results.

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 C 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 E.

Metrics Given our emphasis on *precise* iterative reasoning, we primarily evaluate performance using exact match (EM) between the predicted and gold sets. We also consider F1 score and Jaccard index, which are highly correlated with EM (Spearman's $r = 0.92$) and differ from EM slightly (see Appendix B.2). We additionally report output token counts as a proxy for reasoning efficiency.

3.2 No Tool Use

Table 1 shows the aggregated tool-less results using both the implicit (default) and explicit constraints.

Despite its low construction cost, Time Puzzles is challenging for tool-less LLMs. Our dataset is a *proof of concept* built entirely from easy-to-search factual knowledge that predates each model's training cutoff or release date. Yet even GPT-5 achieves only 49.3% EM on the dataset, with all other models below 31% EM. Providing explicit dates instead of implicit factual constraints improves performance, but a clear **reasoning gap** remains: GPT-5 reaches the best EM at 80.7% and it takes 685B parameters for DeepSeek V3.2-R to get only 77.0% EM. Moreover, model performance tends to decrease as the number of solutions grow (Figure 2).

	Exact Match (%)		Output Tokens	
	Implicit	Explicit	Implicit	Explicit
GPT-5	49.3	80.7 (+31.4)	4382.4	3416.9 (-965.5)
GPT-5-nano	14.0	57.2 (+43.2)	5271.0	4903.9 (-367.1)
GPT-4.1	30.0	58.8 (+28.8)	671.6	593.0 (-78.6)
GPT-4.1-nano	8.2	40.0 (+31.8)	1116.9	1023.6 (-93.3)
GPT-oss-20B	10.2	43.8 (+33.6)	5601.0	4965.2 (-635.8)
DeepSeek-V3.2	27.0	68.5 (+41.5)	785.4	704.6 (-80.8)
DeepSeek-V3.2-R	33.3	77.0 (+43.7)	8276.5	5821.8 (-2454.7)
Qwen3-4B-Inst	2.7	21.2 (+18.5)	2621.0	2563.5 (-57.5)
Qwen3-4B-Think	5.5	38.7 (+33.2)	9529.9	10037.3 (+507.4)
Qwen3-8B	10.0	42.7 (+32.7)	7603.6	7161.4 (-442.2)
Qwen3-14B	11.3	46.5 (+35.2)	4879.9	4782.3 (-97.6)
Ministral-3-8B-Inst	3.7	12.3 (+8.6)	1801.0	1650.5 (-150.5)
Ministral-3-8B-R	3.8	18.7 (+14.9)	2117.6	1749.4 (-368.2)

Table 1: Exact-match accuracy and average output tokens, averaged over six solution counts, for implicit (default) and explicit constraints in the tool-less setting.

Time Puzzles effectively distinguishes nuanced iterative temporal reasoning capabilities. In terms of EM, 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.

Success on Time Puzzles hinges on sustained, structured temporal reasoning rather than exhaustive generation. For example, GPT-5 outperforms all other reasoning models under both implicit and explicit constraints while producing shorter outputs (except the low-performing Ministral-3-8B-R). Similarly, larger Qwen3 models perform better with much fewer output tokens.

3.3 Web Search

Setup We investigate if web search can close the performance gap between the implicit and explicit constraints. We consider six LLMs: GPT-5, GPT-

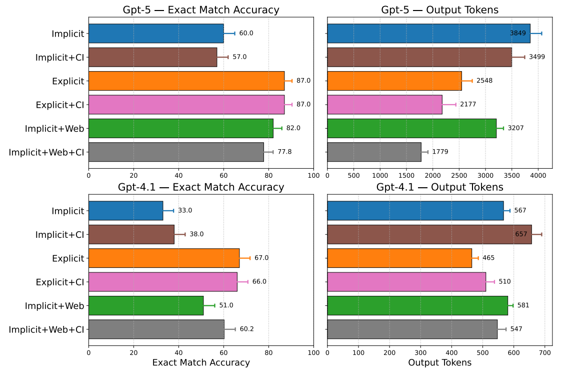

Figure 3: Exact-match accuracy (left) and output tokens (right) for GPT-5 (top) and GPT-4.1 (bottom) under different conditions for single-solution puzzles. Numbers annotated on bars are mean values, beside error bars. `+Web` enables web search. `+CI` enables Code Interpreter.

4.1, GPT-oss-20B, DeepSeek-V3.2-R, Qwen3-14B, and Ministral-3-8B-R. We run GPT-5 and GPT-4.1 with live web search to represent two paradigms: static, one-off retrieval (GPT-4.1) and agentic, iterative retrieval (GPT-5). We then re-use the search results used in the responses of these two models as fixed context for the open-weight LLMs. For cost reasons, we only run web search on a subset of our dataset with solution counts 1, 3, and 5.

Results Figure 2 shows that web search consistently improves performance across different solution counts, but does not close the performance gap between implicit and explicit constraints. Moreover, the marginal gains decrease as the solution count increases. Notably, GPT-5-style web search yields larger improvements than GPT-4.1-style web search, suggesting that iterative temporal reasoning benefits more from agentic, repeated retrieval than from a single retrieval pass.

Overall, these results indicate that even with access to web search, state-of-the-art LLMs still struggle to reliably retrieve and, more importantly, integrate evidence to solve iterative temporal reasoning problems and fully resolve temporal constraints.

3.4 Code Interpreter

Setup We evaluate whether enabling Code Interpreter (+CI) improves performance and/or reasoning efficiency under the previously discussed conditions. For cost reasons, we test only GPT-5 and GPT-4.1 on single-solution puzzles, representing static and agentic uses of +CI, respectively.

Results Figure 3 shows that enabling Code Interpreter (+CI) degrades GPT-5 performance on implicit constraints, both with and without web search, whereas GPT-4.1 benefits from +CI, particularly when combined with web search. In contrast, performance on explicit constraints remains largely unchanged for both models. Moreover, +CI consistently reduces GPT-5 output length, but does not yield similar verbosity reductions for GPT-4.1. Overall, +CI is not a reliable substitute for resolving implicit temporal constraints, though it can improve reasoning efficiency and complement web search for capable non-reasoning models.

4 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.

5 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 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 diagnostic for tool-augmented iterative temporal reasoning and can be systematically extended to support more challenging evaluations.

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.

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

As mentioned in Section 2.2, 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, indicating the granularity of the information provided.

Table 2 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:

$$IG(t) = \log_2(|\mathcal{D}|) - \log_2(|\mathcal{C}(t)|) \qquad (2)$$

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 2, ensures diversity by selecting facts from different categories and balanced constraint levels. The solution verification process, detailed in Algorithm 3, optimizes performance by first filtering with facts with high IG.

Algorithm 2: GENERATEFACT Procedure

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

```
F ← {t_kb};
UsedCats ← {Cat(t_kb)};
// Iteratively generate N-1 distinct facts
for i ← 1 to N − 1 do
    // Ensure balanced levels
    TargetLevel ← DetermineLevel(i, N);
    CandidateCats ← {k ∈ K | k ∉ UsedCats, Level(k) == TargetLevel};
    k_new ← RandomSelect(CandidateCats);
    t_new ← InstantiateFact(k_new, d_seed);
    F ← F ∪ {t_new};
    UsedCats ← UsedCats ∪ {k_new};
end
return F;
```

Algorithm 3: 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 ∈ F;
F_sorted ← SortDescending(F, key = IG);
A ← D;
for t ∈ F_sorted do
    A ← A ∩ C(t);
end
return A;
```

B Evaluation Metrics

In the paper, we primarily use exact match accuracy to measure model performance to emphasize our focus on precise iterative reasoning, but we also consider F1 and Jaccard Index to enhance our evaluation. We describe their computation and the empirical difference between exact match accuracy and F1/Jaccard Index.

B.1 Metric Computation

Let Y_i denote the gold solution set and $\hat{Y}_i$ the predicted set for instance i.

Exact Match Accuracy

$$\mathrm{EM}_i = \mathbb{I}\left[Y_i = \hat{Y}_i\right] \qquad (3)$$

Precision, Recall, and F1

$$\mathrm{Precision}_i = \frac{|Y_i \cap \hat{Y}_i|}{|\hat{Y}_i|}, \quad \mathrm{Recall}_i = \frac{|Y_i \cap \hat{Y}_i|}{|Y_i|}, \qquad (4)$$

$$\mathrm{F1}_i = \frac{2\,\mathrm{Precision}_i\,\mathrm{Recall}_i}{\mathrm{Precision}_i + \mathrm{Recall}_i}. \qquad (5)$$

By convention, when $Y_i = \hat{Y}_i = \emptyset$, all three metrics are set to 1; when exactly one of the sets is empty, all three metrics are set to 0.

Jaccard Index

$$\mathrm{Jaccard}_i = \begin{cases} 1 & \text{if } Y_i = \hat{Y}_i = \emptyset \\ \dfrac{|Y_i \cap \hat{Y}_i|}{|Y_i \cup \hat{Y}_i|} & \text{otherwise} \end{cases} \qquad (6)$$

Final reported scores are computed via macro-averaging:

$$\mathrm{Metric} = \frac{1}{N}\sum_{i=1}^{N} \mathrm{Metric}_i, \qquad (7)$$

where N is the number of evaluation instances.

B.2 Exact Match Versus F1/Jaccard Index

Table 3 compares exact match (EM) accuracy with F1 and Jaccard Index (JI). The mean absolute differences between EM and F1 and between EM and JI are 7.07 and 5.40, respectively. The relatively large standard deviations are expected, as EM is binary at the instance level. Nevertheless, in 87.98% of cases, all three metrics agree, indicating that models typically either solve a puzzle completely or fail entirely. Consistent with this observation, Spearman correlations between EM and the other metrics are high ($r = 0.92$).

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., ··Mon 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 2: Taxonomy of temporal facts used in generation, categorized by constraint granularity level.

	Mean	Std Dev	% Equal	Spearman
EM Vs F1	-7.07	20.52	87.98	0.92
EM Vs JI	-5.40	16.41	87.98	0.92

Table 3: Exact match (EM) accuracy versus F1/Jaccard Index (JI) based on all the experiments ran on this study (over 19,000 data points).

C 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*.

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.

D Additional Results

Figure 4 reports average exact match accuracy across solution counts for the remaining models (for which web search is not implemented) under both implicit and explicit constraints.

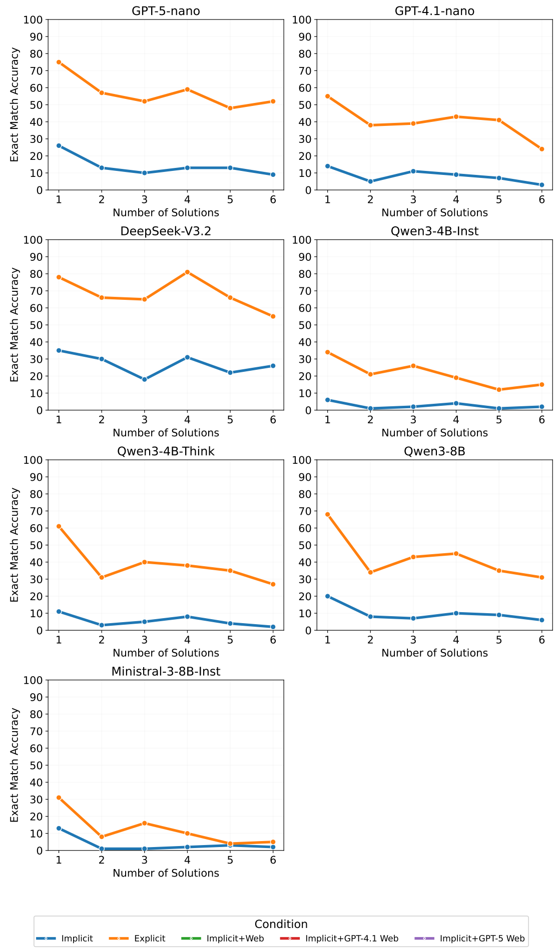

Figure 4: Average exact-match accuracy across solution counts for other models evaluated without web search.

E Prompts

We use the following prompt template. The placeholder `{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".
```