Do Large Language Models (LLMs) Understand Chronology?

Pattaraphon Kenny Wongchamcharoen[1], **Paul Glasserman**[2]

[1] Industrial Engineering & Operations Research, University of California, Berkeley
[2] Columbia Business School, Columbia University
pattaraphon.kenny@berkeley.edu, pg20@gsb.columbia.edu,

Abstract

Large language models (LLMs) have shown great potential as forecasting tools in finance and economics, but backtesting performance is subject to look-ahead bias if the backtesting period overlaps with an LLM's training window. Prompt-based attempts to avoid look-ahead bias require that LLMs understand *chronology*. We test LLMs' ability to understand and enforce chronological order in three canonical types of tasks, each addressing different dimensions of problem complexity: sorting of randomly shuffled historical events; conditional sorting of events defined by some conditions; and anachronism detection based on intersections of multiple timelines. Our experiments use events that we first confirm are known to the LLM via a knowledge pretest; ensuring that we test chronological understanding on an LLM's pretrained internal knowledge. Across three LLM families—GPT-4.1 (*standard*), GPT-5 (*hybrid-reasoning*), and Claude 3.7 Sonnet (*large-reasoning*, with and without Extended Thinking)—we find that performance degrades rapidly with problem complexity but improves greatly with reasoning models when test-time extended reasoning is enabled. Our findings delineate limits of current LLMs on chronological tasks, providing insights into task complexity, and demonstrate scenarios in which reasoning helps. These patterns are important for the real-time application of LLMs in finance.

Code — https://github.com/kennywong524/chronollm

Introduction

Motivated by applications in forecasting, we study chronological reasoning in large language models (LLMs). We investigate how different dimensions of task complexity affect various measures of LLM performance in tests of chronological reasoning; and we examine where recently developed reasoning modes have the greatest impact. We design our tests around the most fundamental elements of chronological reasoning: working within the constraints imposed by a sequence of events. To focus on reasoning, we construct our experiments around events within the LLMs' training data.

LLMs have shown great potential as forecasting tools for finance and economics. Typical applications ask LLMs to predict the direction of stock prices based on text data, such as news reports, company earnings calls, or analysts' research. Evaluating the performance of any forecasting method requires backtesting the method on historical data. Yet it has also been recognized that the backtesting of LLM forecasts is subject to look-ahead bias ((Glasserman and Lin 2024), Sarkar and Vafa (2024)) when the backtesting period overlaps with the LLM's training window: the LLM may be asked to predict an outcome it has already seen. Leakage of post-event information embedded in the LLM's pre-training corpus can inflate estimated forecast performance and lead to disappointing out-of-sample results.

Various methods have been proposed to try to measure and mitigate this problem. The simplest approach is to limit testing to an LLM's post-training period (as in, e.g., Halawi et al. (2024), Lopez-Lira and Tang (2024)). This approach eliminates look-ahead bias, but it severely limits the testing window, particularly for the most recent models. Glasserman and Lin (2024) test LLM performance in predicting the short-term response of stock prices to news headlines; they find that masking company names in the news headlines can be effective in removing look-ahead bias. This idea has been extended to larger texts in Engelberg et al. (2025), but Sarkar and Vafa (2024) and Lopez-Lira, Tang, and Zhu (2025) find evidence that LLMs can see through anonymization in long documents. Building snapshot models trained using only text available up to fixed dates in the past (Sarkar and Vafa (2024), (He et al. 2025)) provides a secure way to wall off future information, but it is computationally demanding and limited to using training documents with clear time stamps.

From a user's perspective, the most convenient solution would be to wall off future information by instructing an LLM to respond using only information available up to a fixed date. However, Sarkar and Vafa (2024) and Lopez-Lira, Tang, and Zhu (2025) find evidence of leakage in examples of this approach. More basically, a prompt-based instruction like "use only information from before date T" presupposes that an LLM understands what "before T" means and can respond accordingly. So here we step back and ask a more fundamental question: *Do LLMs understand chronology?* Before asking an LLM to avoid leakage of future information in responding to new information, we want to evaluate how well the LLM understands chronological constraints in data on which it has been trained. This issue is fundamental to real-time application of LLMs to financial markets.

Markets incorporate information very quickly, so any misalignment in timing can potentially invalidate the application of LLMs to financial data.

Prior approaches to NLP temporal testbeds (e.g., TimeQA (Chen, Wang, and Wang 2021), TRAM (Wang and Zhao 2024), TimeBench (Chu et al. 2024), and ChronoSense (Islakoglu, Yates, and de Rijke 2025)) seek to isolate and evaluate an LLM's performance on specific components of temporal reasoning—such as *implicit temporal reasoning* and *temporal commonsense*—often using newly crafted, hypothetical scenarios and specially designed instructions to probe narrow inference types. Here we take a different approach, evaluating chronological understanding of *information the LLM already knows*; we use historical facts, which we first confirm the LLM knows. This design probes the LLM's ability to integrate temporal reasoning with the model's internal world view. It is less focused on isolating granular components of temporal reasoning and seeks instead to better assess the model's chronological reasoning ability "in the wild." Because chronology is a precondition for avoiding look-ahead bias in forecasting, these minimal tasks serve as a lower bound: if models fail on basic ordering, they cannot be trusted on more complex temporally-conditioned applications.

We test performance on three task types: *(1) Chronological sorting* (putting shuffled events in chronological order); *(2) Conditional sorting* (selecting events that meet a specified condition and ordering those chronologically); *(3) Anachronism detection* (distinguishing possible vs. impossible events based on historical timing). We evaluate results using Spearman's ρ and Kendall's τ, exact match rates, and Cayley distances. We evaluate both non-reasoning and reasoning models from multiple families (OpenAI's GPT-4.1, the newly released GPT-5 series with various reasoning effort hyperparameters and Anthropic's Claude Sonnet 3.7 with and without extended-thinking). We design our tests around basic historical facts. We use U.S. presidents as our primary domain because their timelines are widely known, unambiguous, and reliably stored in LLMs' pretraining corpora. This allows us to isolate chronological reasoning independent of fact retrieval, whereas less-documented historical domains would confound missing knowledge with reasoning errors. This approach makes our tests transparent and easily reproducible. It also makes it simple to verify that the basic facts in question are "known" to an LLM and thus to focus on evaluating chronological understanding. The three categories of tests above allow us to vary problem complexity along distinct dimensions (such as list length, filtering difficulty, and combinatorial combinations).

Our main finding is that LLM performance on our chronology tasks degrades quickly with problem complexity. Interestingly, poor performance at high complexity kicks in at problem scales much smaller than those seen in other tasks (Shojaee et al. (2025)), suggesting that reasoning about the chronology of real events may be inherently difficult. This finding does not bode well for prompt-based mitigation of look-ahead bias using current models, as the complexity of walling off future information for a forecasting task is much greater than the complexity of our experiments. However, performance improves substantially for models with extended reasoning, which achieve near flawless results on both unconditional and conditional sorting tasks.

Our results include several novel findings: (i) Exact match ordering collapses as lists grow, even while rank correlations stay fairly high, with models often dropping or adding historical events to prompted lists; (ii) reasoning models consistently outperform base models across all three task families and all the evaluation metrics we study, and their performance generally increases with larger test-time reasoning budgets; (iii) reasoning models perform slightly better in conditional sorting than when they are simply given a randomly shuffled list; and (iv) a basic anachronism detection task defined by two events proves easy for the LLMs, but performance declines on tests defined by the intersection of multiple events.

Experimental Design and Key Results

In this section, we briefly outline the experimental methodology and present the key results; additional experiments, ablations, and implementation details are provided in the Supplementary Material.

1. Basic Sorting

Our first test presents the LLM with a randomly shuffled subset of U.S. presidents and asks the LLM to order them chronologically by when they served. (We exclude Grover Cleveland and Donald Trump because they served non-consecutive terms.) We first confirm that the LLM knows the dates of the presidents' terms. To do this, we first perform a knowledge-verification step: for each candidate item i, the model must output the year $\hat{y}_i$ that exactly matches the correct y_i; only items passing this check are retained.

Then, we present lists of lengths 2–43 to examine the impact of task complexity. We use several measures of performance at each n. The exact match rate is the fraction of times the LLM returns a perfectly ordered list. Spearman's ρ, Kendall's τ, and the Cayley distance are calculated by comparing the LLM's response with the ground truth ordering; they provide information on how closely the LLM's response approximates the true ordering when it fails to produce an exact match.

Figure 1 plots the frequency with which LLMs sort the list perfectly (Exact match rates). For both GPT-4.1 (black) and Claude-3.7 Sonnet (red), these exact match rates drop sharply for lists as short as 5 or 10, suggesting that this simple task proves surprisingly difficult. We also observed similar performance with GPT-5 at minimal/low reasoning effort, its non-reasoning variant. Interestingly, both GPT-4.1 and Claude-3.7 Sonnet show U-shaped performance for Spearman's ρ and Kendall's τ, indicating that task complexity is not a simple function of list length: the LLMs do somewhat better on a complete set of presidents than on a random subset.

We also observed that non-reasoning models often erroneously omit or add extra presidents names to their responded list, with increasing prevalence at larger list sizes. The blue lines in Figure 1 show the performance of Claude-3.7 Sonnet with Extended Thinking (ET). Remarkably, this

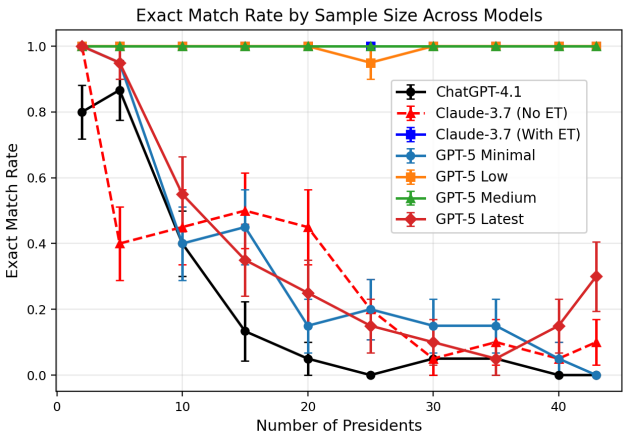

Figure 1: EM rate by list size across various reasoning model families. Error bars are ± 2 s.e. around mean.

reasoning model achieves *perfect* chronological ordering across all list sizes; we also observed the same performance with GPT-5 at *medium/high* reasoning effort. The added test-time deliberation, acting as a private scratchpad for the reasoning models, yields dramatic improvement in understanding a basic notion of chronology.

2. Conditional Sorting

We extend beyond direct ordering by turning to tasks in which the events to be ordered are defined implicitly through some condition (e.g., "born in Ohio or Virginia - OHIOORVIRGINIA"). We hypothesize that a *two-step* prompt—urging the model to *think* by first *filtering* the candidate set before ordering it—may nudge the LLM into a more deliberative reasoning and lower its error rate. Our central question is straightforward: does the LLM sort more accurately when it *discovers* the relevant names itself, or when it is simply *given* the correct subset and asked to order it? Requiring the model to filter events by condition raises complexity, yet the induced reasoning may act as scaffolding that provides LLMs more information regarding the selected items relative to a random list. The conditional sorting experiments test these competing hypotheses.

Figure 2 describes our conditional sorting pipeline. We run paired trials on the same shuffle under two regimes: (i) *self-filtering & sorting* (the model filters events based on the condition, then orders them chronologically) and (ii) *given-names & sorting* (the model just chronologically orders the provided filtered events, presented in the same shuffled order as (i)). We give an example to illustrate the algorithm. Let the shuffled list be $[F, B, D, G, H, E, C, A]$ and $c =$ "born in Ohio or Virginia", with ground truth $G_c = \{B, E, G\}$. In Self-filtering & sorting, we input the full list and instruct "filter for Ohio or Virginia, then order chronologically." The correct LLM behavior will be to filter the full list to $[B, G, E]$ and reorder to $[B, E, G]$ as the final output. For the Given-names & sorting task, we only input $[B, G, E]$ and instruct the LLM to "order the list chronologically." The correct expected output should then be $[B, E, G]$.

Filtering accuracy (OHIOORVIRGINIA)				
Model	Acc.	n_{correct}	n_{total}	ET?
Claude 3.7 Sonnet	0.02	2	100	No
Claude 3.7 Sonnet	0.98	98	100	Yes
GPT-4.1	0.00	0	100	N/A
GPT-5 (medium)	1.00	100	100	N/A
Ordering metrics (Claude 3.7 Sonnet + ET)				
Metric	Value	n_{correct}	n_{total}	Cond.
Spearman's ρ	0.997	99	100	Given
Kendall's τ	0.995	99	100	Given
Exact match	0.82	82	100	Given
Spearman's ρ	1.000	100	100	Self
Kendall's τ	1.000	100	100	Self
Exact match	0.97	97	100	Self

Table 1: Conditional sorting on OHIOORVIRGINIA: filtering accuracy across models and ordering performance for Claude 3.7 Sonnet with Extended Thinking (ET).

For a clean comparison, we keep only self-filtering trials where the model's filtered set matches ground truth; otherwise the filter is marked *invalid* and the trial is excluded from the ordering analysis (duplicates are ignored at filtering but counted as ordering errors later). Each trial begins from one shuffle of all 43 presidents, and two matched conditions are evaluated on that same shuffle.

Table 1 illustrates the result of the conditional sorting task. To our surprise, non-reasoning GPT-4.1 failed to filter correctly, never identifying the 15 presidents from Ohio or Virginia—yielding no valid trials for ordering. In contrast, reasoning models performed nearly perfectly: Claude 3.7 Sonnet with Extended Thinking (ET) and GPT-5 (medium reasoning) both achieved near- or full-accuracy filtering, enabling direct ordering comparisons. With ET, Claude 3.7 Sonnet reached almost identical ordering performance across tasks, and even slightly *outperformed* its given-names baseline on OHIOORVIRGINIA (0.97 vs. 0.82 exact-match). GPT-5 (medium) achieved perfect filtering and ordering overall. Requiring the LLM to select the relevant events before sorting them adds to overall task complexity. Yet the more active involvement of the LLM in selecting the events improves performance in ordering these events. In short, conditional sorting adds difficulty primarily through a filtering bottleneck; once overcome (via ET or higher reasoning effort), ordering is near-perfect and may even benefit from the model's intermediate "thinking" step induced by self-filtering tasks.

We also analyze per-position accuracy for Claude 3.7 Sonnet on given-name tasks, restricting to trials with perfect filtering and equal list lengths (pure ordering errors only). We report the trials with no Extended Thinking (ET attains near-ceiling accuracy) and omit self-filtered runs, where residual errors are largely driven by length mismatches rather than mis-ordering. Per-position accuracy measures how often the model gets each specific rank correct. For a

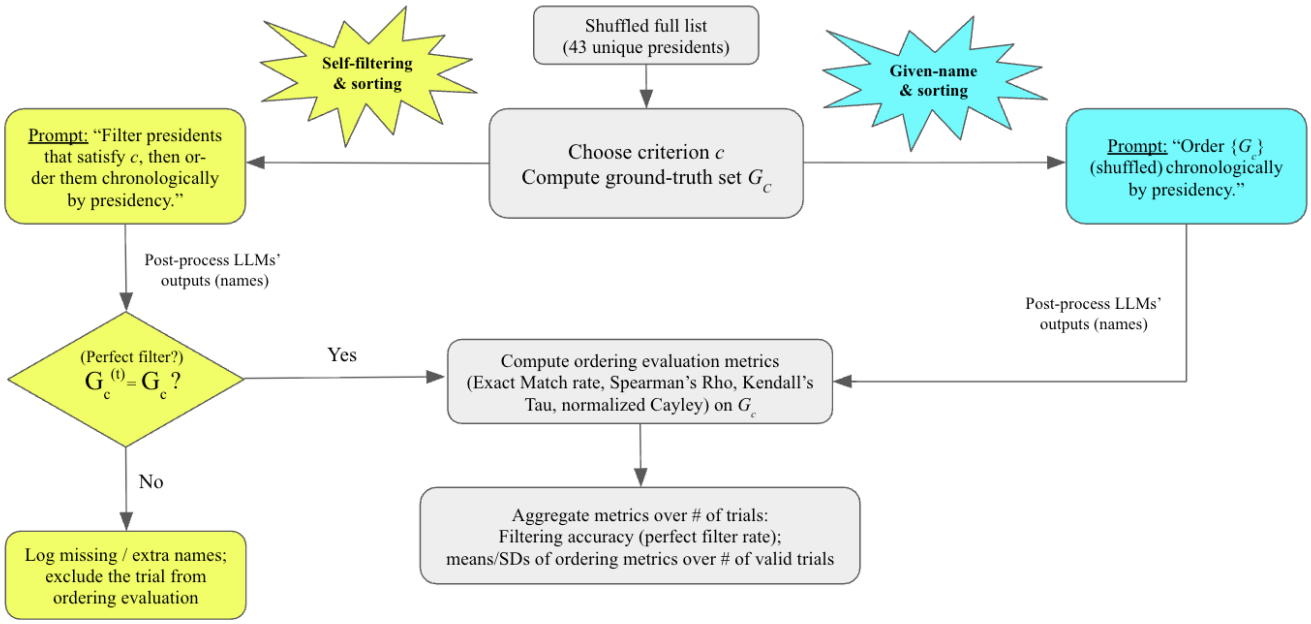

Figure 2: Conditional sorting pipeline: *self-filtering & sorting* vs. *given-names & sorting*. Only trials with $G_c^{(t)} = G_c$ on the left enter the paired comparison. Duplicates count toward ordering errors but not the filtering decision.

given rank r, we compute the share of trials (restricted to equal-length predictions) where the item that should appear at position r is indeed placed at position r. Values lie between 0 and 1, with larger values indicating greater accuracy at that position (see Supplementary Material for full formalization). Figure 3 shows a consistent U-shaped trend: near-perfect accuracy at the first few positions, a trough mid-list, and recovery at the tail. This pattern is aligned with endpoint anchoring—salient earliest and latest figures are placed correctly—while densely packed mid-century presidencies are more frequently confused by the LLM.

In short, conditional sorting adds difficulty primarily through a filtering bottleneck; once overcome (via ET or higher reasoning effort), ordering is near-perfect and may even benefit from the model's intermediate "thinking" step induced by self-filtering tasks.

3. Anachronism Detection

In addition to simply ordering items or conditionally sorting them, we increase the task difficulty by asking the LLM whether a scenario is chronologically possible (e.g., could president A meet president B or use technology X while in office), which requires retrieving and aligning overlapping timelines. Anachronisms can be understood as non-overlapping intervals in the Allen relations studied in Islakoglu, Yates, and de Rijke (2025), but there are important differences between our tests and those in Islakoglu, Yates, and de Rijke (2025). We do not probe individual Allen relations but rather require the LLM to determine whether any of the relations that would make an event possible actually hold. Also, our multi-timeline tests require an LLM to check for the intersection of multiple intervals, rather than just two.

We test anachronism detection via two variants: (i) *single–boundary feasibility*, labeling a president–event pair (p, e) as possible iff the term window $[a(p), b(p)]$ intersects the event's availability interval $[t_{\min}(e), t_{\max}(e)]$; and (ii) *multi–timeline overlap*, asking "were [n specific presidents] $p_1, p_2, ..., p_n$ all alive at the same time?" which require checking $\bigcap_k [\, \ell(p_k), d(p_k) \,] \neq \varnothing$, where $\ell(\cdot), d(\cdot)$ are birth/death years. Ground truth is then built from curated events with first–possible dates.

Many technologies exhibit a lag between invention and reliable in-office or commercialized use. To avoid ambiguous labels, each event e may include a *grey zone* interval $\mathcal{G}(e) = [g_{\min}(e), g_{\max}(e)]$ where $g_{\min}(e)$ is the invention date and $g_{\max}(e)$ is the first recorded use in the White House (i.e., by a sitting president). Any president p whose term window $[a(p), b(p)]$ overlaps $\mathcal{G}(e)$ is *excluded for that event* at data-construction time (i.e., the pair (p, e) is dropped rather than labeled true/false). This prevents false positives/negatives that arise from the time discrepancy. This also helps us remove boundary ambiguity about sporadic or ceremonial access surrounding a new technology.

We run GPT–4.1, requiring bare `Possible`/`Not Possible` outputs, deduplicate repeated pairs, and compute Accuracy, Precision, Recall, and F1 (treating Not possible as the positive class). We process the results by eliminating duplicate events to ensure each unique president-event pair is only counted once. This is necessary because our sampling method allows the same president-event pair to ap-

Batch Size	Type	Acc.	Prec.	Rec.	F1
N=6	3T–3F	0.998	1.000	1.000	1.000
	6F	0.995	N/A	N/A	N/A
	6T	0.995	1.000	0.974	0.987
N=10	10F	0.950	N/A	N/A	N/A
	10T	0.908	1.000	0.890	0.942
	5T–5F	0.932	0.933	0.933	0.933
N=20	10T–10F	0.992	1.000	0.989	0.994
	20F	0.982	N/A	N/A	N/A
	20T	0.982	1.000	0.983	0.991

Table 2: GPT-4.1 anachronism detection results for Variant 1 across batch size and type; "N/A" indicates cases where precision, recall, and F1 are undefined because there are no positive predictions (all-false batches). These batches can still achieve high accuracy (e.g., > 0.95) but cannot be evaluated on precision/recall metrics.

Complexity	Type	Acc.	Prec.	Rec.	F1
43C2 (2 pres.)	10F	0.950	N/A	N/A	N/A
	10T	0.908	1.000	0.890	0.942
	5T–5F	0.932	0.933	0.933	0.933
43C3 (3 pres.)	10F	0.912	N/A	N/A	N/A
	10T	0.766	1.000	0.750	0.857
	5T–5F	0.914	0.936	0.879	0.906
43C4 (4 pres.)	10F	0.930	N/A	N/A	N/A
	10T	0.656	1.000	0.656	0.793
	5T–5F	0.881	0.954	0.798	0.869

Table 3: GPT-4.1 anachronism detection results for Variant 2 (N=10) across increasing combinatorial complexity, where "43C2–43C4" denote the overlapping timeline of 2-4 presidents. "N/A" indicates undefined precision, recall, and F1 in all-false batches (no positive predictions).

pear in multiple batches, and counting duplicates would artificially inflate our overall performance metrics. After deduplication, we compute all evaluation metrics. We compare performance when the LLM is presented with N sets of possible events, N sets of impossible events, or $N/2$ of each, aggregating metrics over 100 trials per batch type.

Variant 1: Single-boundary feasibility For our single-boundary events we use technological inventions that continue to exist indefinitely. For example, correctly determining whether "James Polk used a telephone while president" describes a possible event requires determining whether Polk's term in office intersects with the period since the telephone was invented.

Table 2 shows Variant 1 results across batch sizes $N \in \{6, 10, 20\}$. Across batch sizes $N \in \{6, 10, 20\}$, we find that anachronism detection attains near-ceiling performance: overall accuracy ranges from 95.0% to 99.8%. Mixed batches (balanced true/false) are the most reliable, typically outperforming all-true or all-false batches; performance is stable even with increasing N, indicating that the model's chronological understanding allows it to detect relatively simple anachronisms well across batch sizes. Consistent with basic and conditional sorting, we performed knowledge verification before any anachronism detection task to make sure that the LLM knows all the invention and adoption dates. Despite that, we observe surprising mistakes—declaring that William Henry Harrison could not have ridden on a railroad and that Barack Obama could have used generative AI while in office—contradicting the LLM's own dates for the relevant events. Overall, however, the LLM is highly accurate for well-defined earliest-feasible dates.

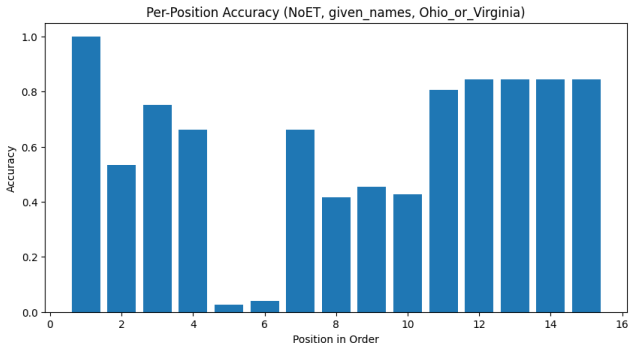

Figure 3: OHIOORVIRGINIA (NoET, given-names). Results over 77 length-matched trials: 2 trials (2.6%) achieved perfect ordering. Accuracy starts at 1.00 near the head, dips mid-list, and recovers to ≥ 0.80 toward the tail.

Variant 2: Multi-timeline overlap We present here one example of our Variant 2 results. We ask the LLM whether a set of 2–4 presidents were ever alive at the same time. This task requires the LLM to check if the given presidents' lifetimes intersect. We compare performance when the LLM is presented with 10 sets that do intersect, 10 that do not, or 5 of each (see Table 3)

We find that accuracy declines as combinatorial complexity increases, though it remains high overall. The model handles pairwise overlaps well but degrades when simultaneously evaluating 3–4 lifespans, pointing to limits in interval reasoning rather than missing facts. We also observe that all-true batches are harder than mixed batches for the LLM, indicating the model is better at spotting impossibilities than verifying large overlapping sets. Consistent with the first variant, increasing prompt batch size (6→10 statements) does not significantly affect performance, suggesting the primary bottleneck is reasoning over increasingly many overlapping intervals, rather than handling a higher number of statements per prompt.

Conclusion

Our experiments show that there are fundamental limitations in LLMs' understanding of chronology. We highlight the following findings in particular. LLM performance on chronological tasks quickly degrades with increasing prob-

lem complexity. Compared with other problem domains, complexity becomes a challenge for chronological tasks even at relatively small scales. There are several dimensions to chronological task complexity, and requiring more "work" (sorting a longer list, adding a filtering step) does not automatically make a task more difficult. Enabling reasoning modes can dramatically improve performance. Without deliberate temporal reasoning and verification, LLMs cannot realize their potential as tools to aid forecasting.

The consequences are practical: without deliberate temporal reasoning and verification, LLMs cannot realize their potential as tools to aid forecasting, as they remain vulnerable to look-ahead bias. Our findings indicate that there is no prompt-only shortcut to eliminating look-ahead bias: if a model cannot reliably handle basic chronological ordering, leakage will persist. But our results also suggest a more promising path through *reasoning models* that explicitly allocate computation to think internally and verify timelines.

Our study has limitations and points to several follow-ups. First, GPT-5's routing to reasoning is itself a challenge. For users who heavily use the chat interface which may or may not auto-route to "thinking" modes, it is unclear whether seemingly simple ordering tasks will be recognized as requiring extra deliberation by ChatGPT 5. Problems will occur if GPT-5 rarely routes to a reasoning mode on its own. Future work should test this explicitly and build an *adaptive* pipeline: run a cheap pass, apply a simple *chronology gate* ("as-of-t" checks, disagreement/uncertainty thresholds), and escalate only when needed (e.g., higher reasoning effort in GPT-5, Extended Thinking in Claude Sonnet).

Second, our model coverage is incomplete, despite our selection of models that represent broad coverage of the types available in the market: non-reasoning (GPT-4.1), reasoning (Claude 3.7 Sonnet), reasoning with internal CoT (Claude 3.7 Sonnet with Extended thinking), or even all-in-one model like GPT-5. We tested a subset of GPT-5 settings and did not systematically evaluate open-source reasoning models. A broader sweep across families (e.g., Llama Mixtral-class, and other open "reasoning" variants) should report *cost–accuracy* trade-offs and when escalation actually enhances performance.

Third, although our tasks are designed to test domain-agnostic abilities, much of the evaluation relies on historical events; adding domain-specific timelines for specific applications or fields (finance, science, multilingual), as well as cross-domain checks will test generalizability.

Finally, beyond single-pass outputs, consensus methods such as LLM-as-judge, self-consistency/majority vote may be helpful in boosting performance. Practical controls should therefore pair time-sliced retrieval prompting or data "fences" with explicit deliberate reasoning. Concretely, we suggest that users should: (i) *utilize reasoning modes* for chronology-sensitive tasks (e.g., GPT-5 with higher reasoning effort, Claude Sonnet with extended thinking), and require the model to enumerate and check timelines before answering; (ii) *add a chronology gate* that asks the model to justify that each fact was knowable as of a specific date t, and (iii) *evaluate its decisions* with other reasoning models (LLMs as a judge) or self-consistency/majority vote over multiple runs and models, including auditing with our chronology experiments.

More broadly during pretraining, building chronologically consistent models will likely require objectives and infrastructure that encode time such as a temporally indexed corpus of data for training LLMs, timeline-aware constraints/regularizers, uncertainty flags/responses for "not knowable as of time t" and robust post-training finetuning tests on feasibility, ordering, and overlap, so that instructions like "answer as if it were 2016" are not just arbitrarily prompted but mechanistically respected by the LLM.

Acknowledgements

This work was conducted as part of Pattaraphon Kenny Wongchamcharoen's summer research internship at Columbia Business School. We gratefully acknowledge the AI in Business Initiative for their generous support, including computational resources and access to the OpenAI API, which made this research possible.

References

Chen, W.; Wang, X.; and Wang, W. Y. 2021. TimeQA: A Dataset for Answering Time-Sensitive Questions. In *Proceedings of NeurIPS 2021*.

Chu, Z.; Chen, J.; Chen, Q.; Yu, W.; Wang, H.; Liu, M.; and Qin, B. 2024. TimeBench: A Comprehensive Evaluation of Temporal Reasoning Abilities in LLMs. In *Proceedings of ACL 2024*.

Engelberg, J.; Manela, A.; Mullins, W.; and Vulicevic, L. 2025. Entity Neutering. SSRN 5182756.

Glasserman, P.; and Lin, C. 2024. Assessing Look-Ahead Bias in Stock Return Predictions Generated by GPT Sentiment Analysis. *Journal of Financial Data Science*, 6(1).

Halawi, D.; Zhang, F.; Chen, Y.; and Steinhardt, J. 2024. Approaching Human-Level Forecasting with Language Models. arXiv:2402.18563.

He, S.; Lv, L.; Manela, A.; and Wu, J. 2025. Chronologically Consistent Large Language Models. arXiv:2502.21206.

Islakoglu, D. S.; Yates, A.; and de Rijke, M. 2025. ChronoSense: Exploring Temporal Understanding in LLMs. In *Proceedings of ACL 2025: Short Papers*.

Lopez-Lira, A.; and Tang, Y. 2024. Can ChatGPT Forecast Stock Price Movements? Return Predictability and Large Language Models. arXiv:2304.07619.

Lopez-Lira, A.; Tang, Y.; and Zhu, M. 2025. The Memorization Problem: Can We Trust LLMs' Economic Forecasts? arXiv:2504.14765.

Sarkar, S. K.; and Vafa, K. 2024. Lookahead Bias in Pretrained Language Models. SSRN 4754678.

Shojaee, S.; et al. 2025. The Illusion of Thinking: Understanding the Strengths and Limitations of Reasoning Models via the Lens of Problem Complexity. arXiv:2506.06941.

Wang, Y.; and Zhao, Y. 2024. TRAM: Benchmarking Temporal Reasoning for Large Language Models. arXiv:2310.00835.