TOWARDS SMARTER LLM-POWERED PREDICTION SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent research has begun incorporating Large Language Models (LLMs) into various predictive modeling tasks. LLMs possess both a plethora of world knowledge from pre-training as well as the ability to receive and use new information in context. This makes them an attractive candidate for next event prediction (Temporal Point Process Forecasting) as they often possess significant prior knowledge on the domain of interest and allow for flexible and non-lossy incorporation of predictive features. In this work we accomplish two main goals. The first is to demonstrate that some of this world knowledge which gives LLMs an advantage over traditional models can be distilled into more lightweight 'custom' traditional models by using an LLM as a model builder. The second is to directly compare various LLM based approaches for predictive modeling in order to understand their various strengths and weaknesses. Despite many recent works introducing various new methods, a proper comparison of different approaches is surprisingly lacking. We evaluate 3 no-finetuning methods along with 2 finetuning based methods across 7 TPP datasets and provide extensive discussion on the various benefits of each approach and design considerations when building LLM-powered prediction systems.

1 INTRODUCTION

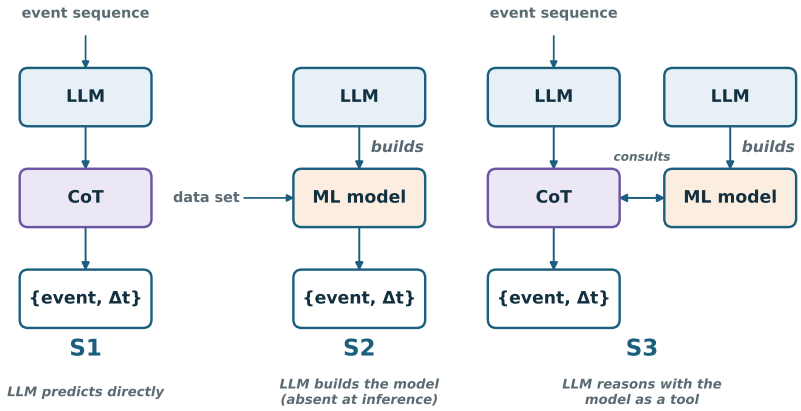
There has been a surge of recent research exploring various usages of Large Language Models (LLMs) for prediction tasks. These tasks extend across a broad range of different domains including time series (Gruver et al., 2023; Jin et al., 2024; Zhou et al., 2023; Cao et al., 2024; Xue & Salim, 2023; Tan et al., 2024; Ansari et al., 2024; Chen & Bibi, 2026; Liao et al., 2026), tabular data (Hegselmann et al., 2023; Hollmann et al., 2023; Han et al., 2024; Abhyankar et al., 2025), recommender systems (Xi et al., 2024; Zhao et al., 2025), finance (Lopez-Lira & Tang, 2023; Guo & Hauptmann, 2024) and more (Cui et al., 2024; Zhu et al., 2024; Zheng et al., 2025; Liang et al., 2024; Feng et al., 2025; Hu et al., 2026; Du et al., 2025; Gong et al., 2025). Another task of particular interest which has recently explored the usage of LLMs is next event prediction, or Temporal Point Process (TPP) forecasting (Gupta et al., 2025). A TPP is a stochastic process of the form $\{(e_i, t_i, f_i)\}_{i\geq 1}$ with $e_i \in E$ a discrete set of event types, $t_i \in \mathbb{R}^+$ denoting inter-event arrival time and $f_i \in F$ denoting additional features / information associated with e_i. The space F is an abstract space here which may contain various numeric and non-numeric features. The task at hand is forecasting; given access to a sample of n iid sequences: $\{\{(e_i^k, t_i^k, f_i^k)\}_{i\geq 1}\}_{k=1}^n$, fit a model to forecast the next event and arrival time (e_{h+1}^k, t_{h+1}^k), conditional on the past event history: $\{(e_i^k, t_i^k, f_i^k)\}_{i=1}^h$. Temporal Point Processes are a general class of processes which are frequently used to model various real world systems across many domains including finance and healthcare and thus being able to model these processes effectively provides significant value to many. The value of using an LLM to aid or even directly replace a traditional model lies in the LLM's world model. LLMs have built up a complex understanding of the world including a breadth of domain knowledge, which if harnessed correctly, can result in more accurate predictions. This line of work marks a paradigm shift in modeling as both general and specific domain knowledge can easily and automatically be leveraged to build powerful models without the need to hand craft a specialized model.

The current literature explores many different ways of using LLMs to leverage their world models across various tasks. These include direct prediction (replace your model with an LLM) (Gruver et al., 2023; Gupta et al., 2025; Hegselmann et al., 2023), feature extraction (LLM suggested or crafted features) (Abhyankar et al., 2025; Küken et al., 2024), prior suggestion (LLM suggests prior distributions used for Bayesian modeling) (Riegler et al., 2025; Capstick et al., 2024) and more Valizadeh et al. (2025). When it comes to TPP forecasting, existing approaches using LLMs typically use them out of the box or finetune them directly to forecast future events (Gupta et al., 2025; Liu & Quan, 2024; Kong et al., 2026; Chen et al., 2026; Shi et al., 2023). We do note one recent work Berghaus (2026) uses an LLM in conjunction with an evolutionary search to build an external script for next event prediction. However, their script is built to apply zero shot without information on the specific dataset and hence the approach is not able to take advantage of the LLM's world model and prior domain knowledge to enhance performance.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

In this paper, we conduct a thorough comparison of varying strategies of incorporating LLMs world knowledge into smarter prediction systems. This comparison enhances our understanding of LLM's and their abilities as well as informs us of the strengths and weaknesses of various approaches scattered across the literature in order to inform future researchers and practitioners. In particular, we consider 3 no-fine-tuning set ups using LLMs for modeling, and compare them across 7 Temporal Point Process Datasets within the recently introduced DescribeEvent benchmark (Wu et al., 2026). In addition, we consider 2 finetuning approaches to compare with the non-finetuning approaches. The first set up (without finetuning), S1, is the direct approach, using an LLM as the predictive model. All model inputs are converted to tokens and passed to a model which provides predictions in text. The second set up, S2, is indirect. We give the LLM access to a dataset and have the LLM produce an external model (i.e. a python script) for making predictions. Lastly, the third set up, S3, we combine the first two approaches by allowing the LLM to make direct predictions but giving the LLM access to the external model and necessary context as built by S2 to call as a tool. See figure 1. Contrary to Berghaus (2026), the goal of the S2 set up is to distill the task specific world model knowledge contained in the LLM into an external model. To the best of our knowledge this is the first work which attempts to distill an LLM's world model knowledge into a small external model for TPP forecasting. In addition, we compare the above approaches with parameter efficient finetuning of an LLM in two ways; with and without Chain of Thought (CoT). We also conduct various secondary experiments and provide extensive discussion throughout on the various design considerations when leveraging LLM's world models to build improved prediction systems.

Figure 1: A summary of the 3 no-fine-tune setups we compare.



There are a few research questions we hope to build towards answering in this work. There is already compelling work demonstrating that LLMs are able to leverage their world models for enhanced predictions (Lopez-Lira & Tang, 2023; Xi et al., 2024; Zheng et al., 2025; Qiu et al., 2026; Wu et al., 2026). Given this prior work, our first research question (RQ1) is as follows: Can an LLM's world model knowledge be distilled into a significantly smaller (non-LLM) external model to improve prediction performance without the added cost of using an LLM at inference time? Are current LLM's capable of performing this distillation via custom model building? If this question is answered in the affirmative, the ramifications are obvious since LLMs are expensive and slow. Our results suggest the answer to this question is yes, but with limitations. Our experiments show that frontier LLM's are capable of producing effective external models which capture some of the LLM's world knowledge and perform much stronger than non-LLM based approaches. However, our results suggest that while an external model is capable of solid performance, not all of the LLM's knowledge is distilled and performance gains are left on the table by removing the LLM at inference time. To this end, we introduce the Inference Time Prior Querying (ITPQ) Hypothesis. This hypothesis states that any compression of an LLMs world model into a static (not capable of reasoning) external model will necessarily be lossy resulting in a lower ceiling for prediction accuracy. We discuss this hypothesis in greater detail in appendix A.

Another topic which we build towards an understanding of, is the role of reasoning ('human like thinking' or pseudo logical deduction) in predictive modeling, and its relationship with traditional computational approaches. Historically, predictive models are fit based on data, allowing models to learn patterns and apply those patterns to make future predictions. With the advent of LLMs a new world of possibilities is opened up as models may now use human like reasoning to make predictions akin to the way a human expert might. Our second research question (RQ2) is: when does reasoning provide value beyond traditional computation based approaches and do current methods effectively combine both reasoning abilities and traditional computational approaches? Our results suggest that reasoning is valuable primarily as a means to incorporate information which is relevant to prediction but is not sufficiently covered

in a dataset used to fit a computational model. This shows up in two ways. Firstly, when no training data exists, reasoning can still make use of predictive features out of the box (zero-shot). We also note that in practice, even when training data exists, it is common for distributional shift to occur or for new features (not present in any training set) to become available at inference time. Reasoning can be an effective approach to handle these 'out of distribution' scenarios. The second type of information which reasoning can make use of is the models world knowledge (which is implicit in the models weights). Our results suggest that more of this information is available via reasoning then is distillable into the external model (S2). However, our results also suggest that finetuning (without reasoning) is also an effective way to incorporate this world knowledge, in addition to statistical pattern matching learned from a dataset. However, the abilities to finetune and access additional datasets are not always available.

We summarize our contributions as follows:

- We provide a comprehensive comparison of various frameworks for incorporating LLMs into prediction systems in order to understand the various trade offs and advantages of each approach. In this way, we hope to reconcile various LLM-based approaches spread out across various problems / fields and bring them together under one common umbrella: using LLM's world models to build more effective prediction systems.
- We demonstrate that using LLMs as custom model builders in contrast to models themselves is a valuable, Pareto efficient (resulting in a strong performance / cost tradeoff) strategy for leveraging LLM World Models.
- We demonstrate via experiments that simple supervised fine tuning is an effective way to incorporate both an LLMs prior world knowledge as well as new data driven insights (learned via a dataset), however, there are still limitations with this approach in terms of marrying the capabilities of pattern matching and human-like reasoning for prediction.
- We provide extensive discussion on many important considerations when designing LLM powered prediction systems including: the various ways in which LLMs can add value, the best approaches to harnessing these gains (along with the tradeoffs), the role of CoT and 'human like reasoning' in prediction modeling and the relationship to traditional data driven modeling, and opportunities for future work in this direction.

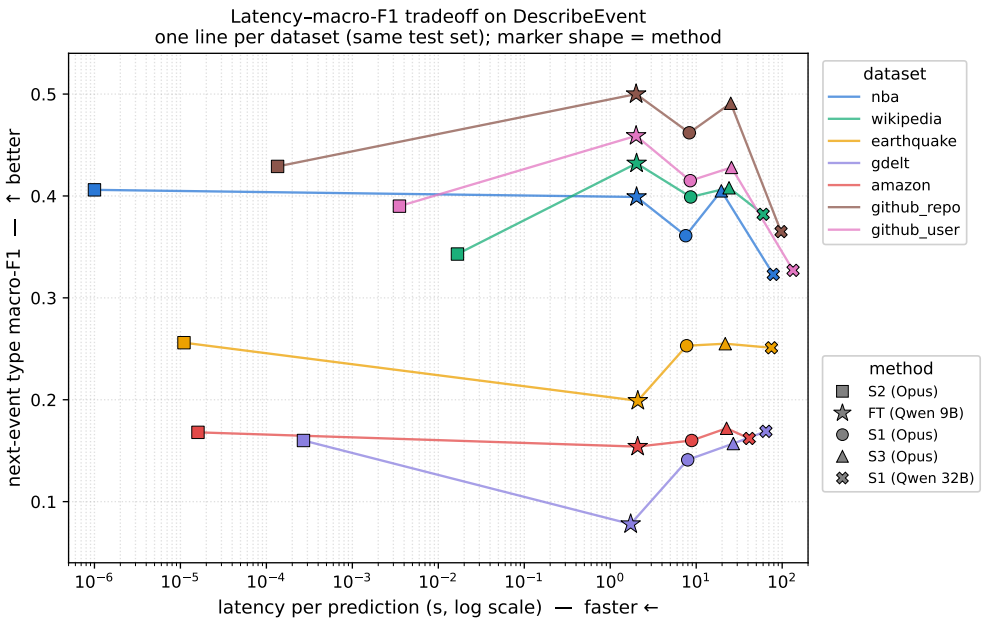


Figure 2: A visual comparison of Macro-F1 and latency across many of the tested approaches. The latency is in seconds and measures average inference time for a single prediction. Build or train time for set ups is not considered here. Important for comparison is hardware used: All opus approaches are run via a Claude Code plan (hardware not disclosed). All QWEN approaches are run on a single A100 40GB GPU. S2 Opus (and the identical model used by Opus in S3) are run on cpu. The same figure with accuracy in place of Macro-F1 can be found in appendix F

2 Related Work

A growing literature shows that large language models supply genuine predictive signal on *quantitative* tasks because pretraining encodes a semantic model of the world, not merely because they are flexible sequence learners. This literature spans finance (Lopez-Lira & Tang, 2023), molecular science (Zheng et al., 2025), clinical prediction (Cui et al., 2024), human mobility (Feng et al., 2025) and more. *How* that prior is brought to bear at prediction time is exactly what our three set-ups separate: the LLM predicts directly at inference (S1) (Gruver et al., 2023; Gupta et al., 2025; Hegselmann et al., 2023); it is used once to distil a static, LLM-free predictor (S2), the "LLM as an automated data scientist" (Hollmann et al., 2023; Hong et al., 2024); or it reasons at inference *with* that model available as a tool (S3) (Chen & Bibi, 2026; Liao et al., 2026). Prior work instantiates each set-up in isolation; we run them head-to-head on a single TPP benchmark. A concurrent evolutionary method (Berghaus, 2026) likewise ships an LLM-free predictor, but as a single zero-shot program applied identically to every dataset, so it cannot exploit the per-domain world knowledge that S2 is built to distil.

Most directly related is the growing use of LLMs specifically for forecasting temporal point processes and event sequences, where the model almost always serves *as* the predictor. Classical neural TPPs parameterise the conditional intensity with recurrent or attention networks (Zuo et al., 2020; Mehrasa et al., 2019); more recent work finetunes or prompts an LLM to forecast event types and inter-arrival times directly (Gupta et al., 2025; Liu & Quan, 2024; Kong et al., 2026; Chen et al., 2026; Shi et al., 2023). We build on the DescribeEvent benchmark (Wu et al., 2026), whose symbolic-event-with-description format our S1 re-implements in its strongest zero-shot configuration. None of this work contrasts a finetuned predictor with a reasoning one on the same task. We defer the broader literature, including how supervised finetuning trades off against in-context reasoning out of distribution, to Appendix B.

3 Three Advantages LLMs Provide for Predictive Modeling

We highlight 3 main ways in which LLMs can provide an edge for predictive tasks, all of which may be entangled but are worth differentiating. We connect these to our research questions. The first is direct signal / prior knowledge. This is information the LLM on the distribution of interest the LLM already possesses. For example, the Point Of Interest (POI) task (which is forecasting a TPP) requires predicting the next venue a user visits based on their recent history and location (Feng et al., 2025). The LLM may know certain venues are closed at certain times and thus can factor this into its prediction of the next venue distribution. This type of signal / prior knowledge is likely inferable from the data and thus the value of this knowledge depends on the amount of data available and the degree to which the model is expected to be able to generalize in production. Specific to the POI task, there is often plentiful data across a large set of venues but limited venue specific data. In addition to our main experiments we demonstrate the value of the S2 approach on a POI task in appendix K. Across all our experiments, we found that this type of knowledge was largely not distillable under S2.

The second way LLMs can add value is as feature encoders. Often feature encoding can be lossy, for example in POI, venue names may be converted to a one hot encoding. However, venue names may contain important information that is stripped during this conversion. For example, for venues which are restaurants, the venue name may provide signal on the type of cuisine. This brings new patterns into reach, for example: 'the user visits a Mexican restaurant every Tuesday' in contrast to just visiting any restaurant on Tuesdays. Additionally, unstructured data such as natural text can be incorporated as additional features such as in the DescribeEvent benchmark (which we use to conduct our experiments). To otherwise incorporate natural text as features without using an LLM, substantial custom modifications may be required to an existing model or handcrafting features, both of which often require significant human effort. In this way, LLMs can harness signal from existing or new features which is otherwise lost (Hollmann et al., 2023; Han et al., 2024). Note that in order for these features to provide new signal, the LLM may need to make use of its world model to understand textual features and how to interpret them in the context of the current problem. In our experiments, we found this to be an effective avenue for distillation under S2. We found that the LLM often built many handcrafted features from the textual descriptions, demonstrating an understanding of how the information contained in the descriptions is relevant for the task at hand.

Lastly, as we previously mentioned, LLMs can provide additional value via adaptability (reasoning). For example, when presented with a context that is unfamiliar (or 'out of distribution' from the training data), the ability to reason about the scenario can provide insight for prediction that no static model could possess without the ability to use 'human like reasoning' and world knowledge of the data generating mechanism to reason over. This reasoning ability is of course not distillable to an external model. We provide additional discussion on the ways in which world model distillation came to fruition under the S2 set up in appendix C.

4 LLM BASED APPROACHES FOR PREDICTIVE MODELING

Our first experiment compares three distinct set ups using LLMs for next event prediction (in addition to a simple finetuning approach). The first set up (S1) makes use of an LLM for direct predictions zero-shot. For this set up, the event sequence history is encoded into a prompt along with a task and dataset description (following Wu et al. (2026)). The LLM produces a prediction in a specified text format from which we extract the final predicted values.

For the second set up (S2) we use Claude CLI to launch an agentic session where the LLM has the ability to write and execute code within a workspace which includes a prompt which once again describes the dataset and the task at hand. Additionally, the workspace contains the dataset in a format explained within the prompt. The task of the LLM is to use the provided dataset to build an external model, ultimately producing a python script 'predict.py' which produces a forecast from provided event history. We emphasize that the LLM should do its best to impart its world knowledge onto this external model.

The third set up (S3) is a combination of the first two set ups. Similar to S1, the model receives the sequence history via a prompt with the same format. In contrast to S1, S3 is operating in an agentic session allowing for the use of tools including writing and executing code and calling the predict.py script produced by S2 (along with the appropriate workspace and context). The instructions prompt for S3 inform the LLM of its access to the existing model and workspace and tell it to both form its own prediction first, then query the existing model and then reason to make its own decision and do any other tool calls necessary (which could include querying the training set or performing other calculations etc). We found that if a more passive prompt was used not instructing the LLM to reason through its own opinion before calling the existing model, it would overly rely on the existing model even when it was capable of making better predictions on its own. The exact prompts used for each approach can be found in appendix H. See figure 3.

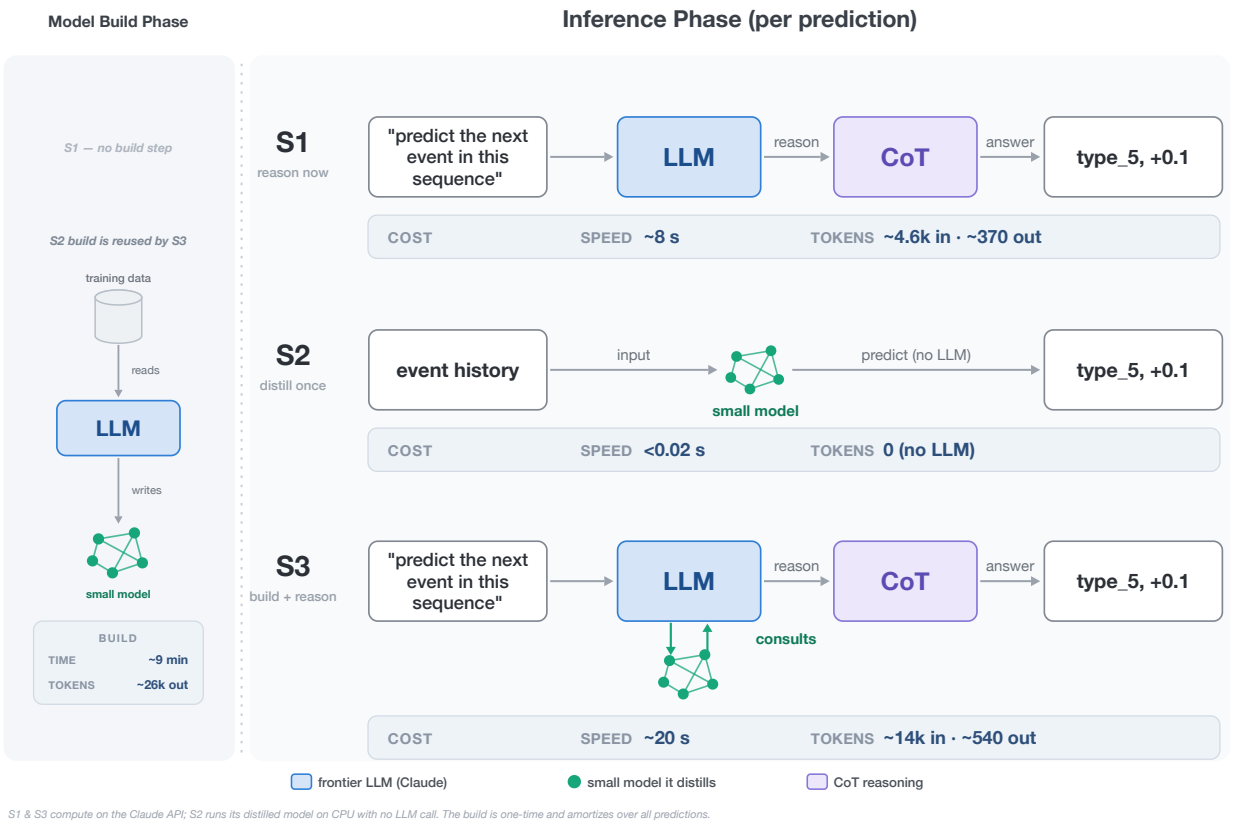


Figure 3: Summary of S1, S2 and S3 and the break down of cost across model building and inference. All three scenarios correspond to Opus 4.8.

4.1 FINETUNING

The obvious limitation of using LLMs directly as in S1, is that any additional data possessed must either be discarded or incorporated in context via the prompt or through finetuning. Incorporation via prompt is an incredibly inefficient way of encoding information and can result in a bottleneck due to context window. Finetuning avoids the reliance on context window and offers the additional benefit of allowing the LLM to learn the necessary computations for the task (pattern matching) internally. However, even parameter efficient finetuning methods such as LORA (Hu et al., 2022), QLORA (Dettmers et al., 2023) and soft prompting (Lester et al., 2021) are often prohibitively expensive (requiring a reduction in model size and hence capacity) and can only be performed on open source models. Finetuning LLMs for downstream applications is common practice and has been explored already in the domain of TPP (Gupta et al., 2025; Liu & Quan, 2024; Kong et al., 2026).

Finetuning in a context like this deserves some special attention. Comparing Wu et al. (2026)'s zero shot implementation and Gupta et al. (2025)'s finetuning, there is an important distinction which is the use of Chain of Thought (CoT) for reasoning. A zero shot CoT reasoning model has the ability to think through any given sequence in detail and run through scenarios before providing an answer, with the limitation that the method is zero-shot and hence cannot learn patterns specific to the unobserved dataset. In contrast, finetuning in the style of Liu & Quan (2024); Gupta et al. (2025); Kong et al. (2026); Chen et al. (2026), does not make use of CoT, since the only labels provided are on the final answer (not the CoT), so the only output of the LLM is the final prediction for event type and arrival time, constraining any 'reasoning' or incorporation of world knowledge to just a few recursive forward passes, in contrast to the open ended-ness of CoT. The benefit of this finetuning is the ability of the LLM to internalize patterns that occur within the data it could not have possibly been aware of apriori and leverage these for prediction. Reasoning without finetuning leans harder on world knowledge and finetuning without reasoning uses insights learned from the data (the standard approach of predictive models) in conjunction with prior world knowledge. The ability to incorporate human like reasoning into a predictive model is a new capability in the era of LLMs and its potential is not fully understood. We therefore conduct two forms of supervised finetuning: a direct finetuning where the model receives the same prompt as in S1 and directly outputs its prediction (in tokens) without any explanation and a second version which we force the model to perform a CoT before providing its answer and then ultimately perform supervised finetuning directly on the answer tokens. This allows us to compare the performance of a finetuned model with the three previous set ups (S1,S2,S3) as well as measure the impact of CoT within a supervised finetuning for event prediction setting. In both fine tuning scenarios we use Qwen 3.5 9B and use LoRA for parameter efficient finetuning.

5 EXPERIMENTS

5.1 DATASETS

We conduct our main experiment on the DescribeEvent benchmark. This benchmark contains 7 TPP datasets spanning various orders of magnitudes of inter-event arrival times and contains a total of 148,185 events, each equipped with a natural language description. These datasets are NBA, wikipedia, earthquake, gdelt, amazon, github repo and github user. We also adopt Wu et al. (2026)'s 'symbolic event type' prompting strategy for S1 and S3. We conduct our experiment using both a leading frontier model, Claude Opus 4.8 and the strongest open source model considered in Wu et al. (2026), Qwen 3.5 32B. We follow their convention for train / test split used training their baseline methods. All Qwen based experiments are conducted on a single 40GB A100 GPU, with each dataset (training and inference) finishing within 8 hours. All Claude Opus experiments were run over the course of 2 months within budget of a Claude Code Max 20x plan.

5.2 RESULTS

Our main results can be found in table 1. There are many interesting observations to be made parsing these results. Firstly, we note that Qwen S1 is effectively a re-implementation of the strongest performing prior method on DescribeEvent and by comparing Opus S1 to Qwen S1 we can observe the large delta in prediction accuracy attributable to a stronger reasoning model, consistent with the thesis of Wu et al. (2026).

Looking at Opus S2, we see that the stronger reasoning LLM is capable of distilling some of its world model into an external model which significantly outperforms Qwen S1 and all non-LLM baselines reported in Wu et al. (2026) (the DE-Base column).This finding is crucially important for practitioners. While using LLMs directly (S1) has been demonstrated by prior works to provide a significant edge over existing methods, the obvious caveat is that in many practical applications it is not reasonable from either a financial or latency perspective to leverage such a large model for this sort of task. Our results demonstrate that a one time upfront cost can be paid by using a frontier LLM to

produce a model which runs LLM free at inference time, harnessing world model gains without paying the price of an LLM at inference time. That being said if we compare Opus S1 and Opus S2, we observe that although S2 consistently outperforms in terms of MAE, it is lagging in terms of Accuracy and Macro F1 for wikipedia, Github User and Github Repo. This suggests that while effective at building an external model and incorporating various aspects of its world knowledge into said model (which is verifiable by examining the workspaces output and notes), not all of Claude's prediction capabilities are being distilled into the external model. A description of the models built under S2 for each dataset can be found in appendix I. This comparison provides the answer to our first research question (RQ1) and we consider this as evidence supporting the Inference Time Prior Querying Hypothesis. Additionally, the gap between S1 and S2 on certain datasets, despite S2 having access to training data, suggests that S1 is able to access additional world knowledge implicit in its weights via reasoning to make informed predictions. This observation along with the general competitive performance of S1 despite not making use of training data provides the basis for RQ2. See figure 2 for the Macro-F1-latency tradeoff f the various approaches.

Another important observation is that Qwen S2 significantly under-performs all other approaches. In order for an LLM to succeed at distilling its world knowledge to an external model, it also needs many additional skills in terms of how to design, build and evaluate prediction systems, a complex task requiring strong coding abilities and knowledge of statistics and machine learning. The large gap between Qwen S2 and Opus S2 illuminates the gap between a large closed source frontier model and a much smaller 32B parameter open source model.

We now consider S3. We did not implement S3 for Qwen as it was clear by Qwen's failed S2 performance that at best S3 would reproduce S1. However, for Opus S3, we observe that S2s strong temporal predictions as measured by MAE are mostly inherited by S3. S3 also performs as well as or stronger than S1 in every scenario which suggests that the ability to perform external computations via a separate model can improve the LLMs abilities when it comes to out of the box performance. S3 is effectively able to determine when to rely on an external models computations and when to 'trust its gut' and make the prediction on its own.

We now consider the finetuning results. The main observation is that the performance in terms of accuracy of the finetuned Qwen 3.5 9B model is the strongest reported (highest on 5/7 datasets) superior to the much stronger model, Opus 4.8, used out of the box. In the domains where we suspect prior world knowledge has the largest effect (Github User, Github Repo and Wikipedia, i.e. domains where S1 is stronger than S2), finetuning yields a notable performance gain for both accuracy and F1. In the other domains (Earthquake, GDELT, Amazon and NBA, where S2 is competitive with S1) finetuning was less effective and in some cases trades accuracy for F1 performance. In general, MAE is still dominated by S2 (and inherited by S3). These results suggest that when world knowledge is highly relevant to the task, finetuning is an effective way to integrate this information along with data driven insight learned from a training set.

We report a side by comparison of FT and the CoT conditioned FT approach previously described in appendix E. We note that conditioning the finetuning on CoT provides no clear gain over simple direct fintuning. This suggests that any prior knowledge or 'reasoning' type behavior the model uses to make predictions is incorporated via direct predictions with finetuning and does not require the open ended exploratory reasoning that CoT offers.

5.3 CoT Reasoning for Out of Distribution Inference

A deeper underlying comparison taking place in our experiments is between two fundamentally different approaches to modeling and prediction. On one hand you have the classical approach which incorporates various modeling techniques (such as a model built by S2) and is built to capture patterns in a data set and exploit them for future inference. In contrast, S1 takes a very different approach. There is no learning implicit patterns from a dataset but rather S1 uses CoT and ultimately reasons akin to the way a human expert would about a scenario and uses logical deduction and heuristics or 'intuition' to guide their predictions. We share some examples of the CoT's produced and the corresponding predictions in appendix J. The ability to use this human like reasoning to guide inference opens up a new world of possibilities and begs the question of whether or not human like reasoning fits into the world of forecasting and if so then where? The story our results tell is that in many cases this human like reasoning can result in fairly strong predictions and does not require any training data. The performance of this approach (on the tested datasets) is as strong and often stronger then any non-LLM baseline including models custom built for the task by a Frontier LLM (S2). Further it is clear that a stronger reasoning model results in more accurate predictions. However, better performance can be still be attained by fine-tuning much smaller LLMs. Finetuning allows for the incorporation of the uncompressed world model knowledge simultaneously used in conjunction with data driven pattern matching. Further, our results suggested that including CoT reasoning pre-empting the supervised label predictions does not improve performance over direct predictions (no CoT). We want to be careful not to conflate 'reasoning' with CoT as while CoT has shown to improve 'reasoning' performance for many tasks (Kojima et al., 2022), there is limited

Dataset	Metric	Opus (S1)	Opus (S2)	Opus (S3)	Qwen (S1)	Qwen (S2)	FT	DE-Base
GitHub Repo	Acc↑	0.535	0.496	0.548	0.417	0.265	**0.569**	–
	F1↑	0.462	0.429	0.491	0.365	0.052	**0.500**	0.332
	MAE↓	2.247	2.116	**2.109**	2.266	2.317	2.147	2.288
GitHub User	Acc↑	0.495	0.473	0.502	0.373	0.241	**0.533**	–
	F1↑	0.415	0.390	0.428	0.327	0.079	**0.459**	0.327
	MAE↓	1.485	**1.396**	1.409	1.498	1.890	1.422	1.563
Earthquake	Acc↑	0.382	0.388	0.390	0.361	0.323	**0.404**	–
	F1↑	0.253	**0.256**	0.255	0.251	0.248	0.199	0.175
	MAE↓	0.328	**0.326**	**0.326**	0.359	0.355	0.431	0.335
GDELT	Acc↑	0.222	0.205	0.220	0.214	**0.243**	0.242	–
	F1↑	0.141	0.160	0.157	**0.169**	0.097	0.078	0.153
	MAE↓	9.884	**9.543**	9.607	10.417	11.940	12.135	9.688
NBA	Acc↑	0.582	0.588	0.607	0.572	0.114	**0.629**	–
	F1↑	0.361	**0.406**	0.405	0.323	0.104	0.399	0.362
	MAE↓	0.070	0.061	0.062	0.075	0.913	**0.060**	0.073
Wikipedia	Acc↑	0.389	0.358	0.382	0.367	0.227	**0.414**	–
	F1↑	0.399	0.343	0.408	0.382	0.181	**0.432**	0.256
	MAE↓	16.970	**16.264**	16.362	17.268	17.249	16.594	16.720
Amazon	Acc↑	0.250	0.281	**0.283**	0.250	0.228	0.276	–
	F1↑	0.160	0.168	**0.172**	0.162	0.096	0.154	0.130
	MAE↓	5.691	5.697	**5.665**	5.804	5.827	5.757	5.780

Table 1: Results on DESCRIBEEVENT, FT = Qwen3.5-9B finetuned (LoRA), validation-selected across 3 seeds; S1 = verbal prior (inference-time reasoning); S2 = computational prior (model built in pre-amble, absent at inference); S3 = verbal prior reasoning *with* the S2 model available as a tool at inference (active-calibration prompt). DE-Base is taken to be the per dataset maximum of the 9 baseline, non-LLM approaches implemented in the original DescribeEvent paper (Wu et al., 2026) (accuracy not reported). 5 of the baselines contributed at least one maximum value to the DE-Base column. Qwen (S1) is effectively a re-implementation of the strongest performing LLM and hence method presented in the DescribeEvent paper. MAE reported in each dataset's native time unit. **Bold** = best per dataset/metric. Underline = second best.

research on the effect of CoT in the context of quantitative prediction tasks and there is no evidence that a similar 'reasoning' is not applied internally in the forward passes of the LLM even without CoT.

Given the strong performance of CoT out of the box (but lack of value under finetuning), one avenue where this approach could benefit is in Out of Distribution (OOD) settings such as distribution drift in a production setting or in order to incorporate new features or context at inference time that wasn't present in a training set. In this setting, a trained model is forced to extrapolate out of distribution. To avoid this potentially dangerous extrapolation, human like reasoning may act as a strong alternative in this setting. We provide a comparison between these two approaches under a natural distribution shift, constructed by augmenting the Wikipedia dataset.

Experimental design. Wikipedia articles differ sharply in editorial contention. We split them by their number of distinct editors into a **stable** regime ($\leq$10 editors: calm, low-revert editing) and a **contested** regime ($\geq$25 editors: edit wars, with markedly higher revert and content-removal rates). We finetune the small model (Qwen3.5-9B) on *one* regime and evaluate it on *both*, and compare against the training-free reasoner S1. The finetuning recipe is identical to the main experiments; only the training subset changes. To probe the shift in both directions we run this twice: (i) train on stable, making contested out-of-regime, and (ii) train on contested, making stable out-of-regime. The evaluation set is always the full standard Wikipedia test set, partitioned into the two regimes at scoring time by editor count; S1's predictions are simply re-scored on each partition (no recomputation, and S1 is by construction identical across the two runs). This yields, for each regime, one finetuned model that trained *in* that regime and one that trained *outside* it, against a common training-free reference.

Analysis. We see very clearly that the finetuned model performs better in whichever regime is was trained in and performance drops in the alternate as expected. The interesting part comes when contrasting these results with S1.

Method	Trained on	Macro-F1	
		Stable (≤ 10)	Contested (≥ 25)
S1 (reasoning)	— (training-free)	**0.280**	0.431
FT	stable	0.267	0.410
FT	contested	0.242	**0.436**

Table 2: **Distribution shift on Wikipedia (macro-F1).** Out of its training regime the finetuned model drops below the training-free reasoner S1, in both directions; FT entries are the mean over 3 seeds.

In the stable regime S1 outperforms both scenarios and in the contested regime the FT performance is higher when this regime is in-distribution but lower when it is OOD. This crossover demonstrates that if there is any misalignment between training and deployment distributions this can make out of the box reasoning a more favorable approach.

5.4 Additional Experiments

Beyond the main experiments outlined above, we conduct a few secondary experiments which we present in the appendix. We explore the impact that the magnitude of data has on the S2 scenario by artificially restricting the size of the datasets to observe how performance changes as data becomes more limited. More details and results can be found in appendix G. We also demonstrate the value of the S2 approach by testing it on a specialized, competitive TPP task, POI. Here we demonstrate on a single dataset that S2 is competitive with research grade custom architectures. Details can be found in appendix K.

6 Final Discussion

In this work we contrast various approaches which use LLMs for predictive modeling. These approaches aim to harness the LLM's built up world model to harness additional signal for prediction that traditional models cannot access. Our results highlighting the various strengths and weaknesses of different approaches which can guide practitioners in choosing which approaches are most applicable to their setting. When latency or model serving cost are an important factor we demonstrate that one can still leverage LLMs for model building, constructing custom models that take advantage of the LLM domain knowledge and provide strong performance, particularly in comparison to non-LLM methods (the latency and performance comparison can be found in figure 2). When access to data is limited, a strong reasoning model using CoT and operating out of the box can also provide very strong performance, even exceeding no-LLM-at-inference methods even when those methods have access to training data. Further, a strong reasoning model is capable of determining when to trust an existing model which has learned from the data or when it should override a data driven model and make a prediction itself. Lastly, when data is available and resources permit, finetuning even a much smaller LLM based model remains the strongest method for incorporating the LLMs world knowledge in addition to learning from a dataset.

It may be unsurprising to some that finetuning a smaller model beats a larger out of the box model as this is often the case for many standard LLM tasks (Hsieh et al., 2023). However, predictive modeling for TPPs is not a standard LLM task. We emphasize that this comparison from out of the box, CoT reasoning to an answer versus finetuned to provide a direct answer (without reasoning) is understudied when using LLMs for TPP forecasting. Further, the comparison here is not simply finetuned vs not finetuned but rather a comparison of two fundamentally different approaches to prediction, data driven pattern matching versus human like reasoning. Lastly, our synthetic results suggest that for certain settings when both human like reasoning and data driven pattern matching are necessary for optimal performance, finetuning is insufficient for incorporating both mechanisms. In theory, reinforcement learning can train both data driven pattern matching and CoT reasoning in conjunction (Chu et al., 2025). Though this approach can be significantly more expensive than finetuning and future work is needed to investigate if in practice reinforcement learning actually trains a model to use human like reasoning and data driven pattern matching harmoniously. As a final note we emphasize that the various approaches we explore here have their strengths and weaknesses and the value of these strengths and weaknesses are dependent on the dataset and task at hand. The edge LLMs offer over traditional models depends on the task which our results reflect via the split of datasets; GitHub Repo, GitHub User and Wikipedia benefit much more from having an LLM at inference time.

ETHICS STATEMENT

We do not foresee ethical concerns specific to this work: it does not involve human subjects or new data collection, and the benchmarks used are already public.

REPRODUCIBILITY STATEMENT

We describe our experimental setup, datasets, and evaluation metrics, and include the prompts used for S1, S2, and S3 in Appendix H. Given the sampling-based nature of LLM inference and the finetuning process, we expect our results to be reproducible up to seed-level noise from these descriptions and prompts. We intend to release code in a future public repository.

LLM USAGE DISCLOSURE

We used generative AI tools (Claude) for the implementation of experimental code, for literature search and summarization of related work, and for assistance with drafting and editing the manuscript. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Nikhil Abhyankar, Parshin Shojaee, and Chandan K. Reddy. Llm-fe: Automated feature engineering for tabular data with llms as evolutionary optimizers, 2025. arXiv:2503.14434.

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. Chronos: Learning the language of time series. *Transactions on Machine Learning Research*, 2024.

Lisa P. Argyle, Ethan C. Busby, Nancy Fulda, Joshua R. Gubler, Christopher Rytting, and David Wingate. Out of one, many: Using language models to simulate human samples. *Political Analysis*, 31(3):337–351, 2023. arXiv:2209.06899.

Arjun Ashok et al. Beyond naïve prompting: Strategies for improved context-aided forecasting with large language models. *arXiv preprint arXiv:2508.09904*, 2025.

David Berghaus. EVIL: Evolving interpretable algorithms for zero-shot inference on event sequences and time series with LLMs. *arXiv preprint arXiv:2604.15787*, 2026.

James Brand, Ayelet Israeli, and Donald Ngwe. Using large language models for market research. Working Paper 23-062, Harvard Business School, 2023. SSRN 4395751.

Defu Cao, Furong Jia, Sercan O. Arik, Tomas Pfister, Yixiang Zheng, Wen Ye, and Yan Liu. Tempo: Prompt-based generative pre-trained transformer for time series forecasting. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.04948.

Alexander Capstick, Rahul G Krishnan, and Payam Barnaghi. Autoelicit: Using large language models for expert prior elicitation in predictive modelling. *arXiv preprint arXiv:2411.17284*, 2024.

Bryan Chan, András György, and Dale Schuurmans. Toward understanding in-context vs. in-weight learning. *arXiv preprint arXiv:2410.23042*, 2024.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Madry. Mle-bench: Evaluating machine learning agents on machine learning engineering. In *Proceedings of the 13th International Conference on Learning Representations (ICLR 2025)*, 2025. arXiv:2410.07095.

Stephanie C. Y. Chan et al. Data distributional properties drive emergent in-context learning in transformers. *arXiv preprint arXiv:2205.05055*, 2022.

Edwin Chen and Zulekha Bibi. Machine learning as a tool (mlat): A framework for integrating statistical ml models as callable tools within llm agent workflows, 2026. Gemini 3 Hackathon submission (PitchCraft project); arXiv preprint, not peer reviewed.

Ke Chen, Peiran Wang, Yaoning Yu, Xianyang Zhan, and Haohan Wang. Large language model-based data science agent: A survey, 2025. arXiv:2508.02744.

Lili Chen et al. Enhancing temporal awareness in large language models for temporal point processes. *arXiv preprint arXiv:2601.00845*, 2026.

Yutian Chen, Xingyou Song, Chansoo Lee, Zi Wang, Qiuyi Zhang, David Dohan, Kazuya Kawakami, Greg Kochanski, Arnaud Doucet, Marc'Aurelio Ranzato, Sagi Perel, and Nando de Freitas. Towards learning universal hyperparameter optimizers with transformers. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, 2022. arXiv:2205.13320.

Yizhou Chi et al. Sela: Tree-search enhanced llm agents for automated machine learning. *arXiv preprint arXiv:2410.17238*, 2024.

Tianzhe Chu et al. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*, 2025.

Hejie Cui, Zhuocheng Shen, Jieyu Zhang, Hui Shao, Lianhui Qin, Joyce C. Ho, and Carl Yang. Llms-based few-shot disease predictions using ehr: A novel approach combining predictive agent reasoning and critical agent instruction. In *Proceedings of the AMIA 2024 Annual Symposium*, 2024. arXiv:2403.15464.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*, pp. 10088–10115, 2023. arXiv:2305.14314.

Hongru Du, Yang Zhao, Jianan Zhao, Shaochong Xu, Xihong Lin, Yiran Chen, Lauren M. Gardner, and Hao Frank Yang. Advancing real-time infectious disease forecasting using large language models. *Nature Computational Science*, 5, 2025. doi: 10.1038/s43588-025-00798-6. arXiv:2404.06962.

Xi Fang, Weijie Xu, Fiona Anting Tan, Jiani Zhang, Ziqing Hu, Yanjun Qi, Scott Nickleach, Diego Socolinsky, Srinivasan Sengamedu, and Christos Faloutsos. Large language models (llms) on tabular data: Prediction, generation, and understanding – a survey, 2024. arXiv:2402.17944.

Jie Feng, Yuwei Du, Jie Zhao, and Yong Li. Agentmove: A large language model based agentic framework for zero-shot next location prediction. In *Proceedings of the 2025 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2025)*, 2025. arXiv:2408.13986.

Shanshan Feng, Feiyu Meng, Lisi Chen, Shuo Shang, and Yew Soon Ong. ROTAN: A rotation-based temporal attention network for time-specific next POI recommendation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 759–770, 2024. doi: 10.1145/3637528.3671809.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *International conference on machine learning*, pp. 10764–10799. PMLR, 2023.

Zorik Gekhman et al. Does fine-tuning llms on new knowledge encourage hallucinations? *arXiv preprint arXiv:2405.05904*, 2024.

Chenghua Gong, Rui Sun, Yuhao Zheng, Juyuan Zhang, Tianjun Gu, Liming Pan, and Linyuan Lv. Epillm: Unlocking the potential of large language models in epidemic forecasting, 2025. arXiv:2505.12738.

Antoine Grosnit et al. Large language models orchestrating structured reasoning achieve kaggle grandmaster level. *arXiv preprint arXiv:2411.03562*, 2024.

Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew G Wilson. Large language models are zero-shot time series forecasters. *Advances in neural information processing systems*, 36:19622–19635, 2023.

Siyuan Guo, Cheng Deng, Ying Wen, Hechang Chen, Yi Chang, and Jun Wang. Ds-agent: Automated data science by empowering large language models with case-based reasoning. In *Proceedings of the 41st International Conference on Machine Learning (ICML 2024)*, 2024. arXiv:2402.17453.

Tian Guo and Emmanuel Hauptmann. Fine-tuning large language models for stock return prediction using newsflow. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track (EMNLP 2024)*, 2024. arXiv:2407.18103.

Shubham Gupta, Thibaut Durand, Graham W. Taylor, and Łukasz Białokozowicz. LAST SToP for modeling asynchronous time series. *arXiv preprint arXiv:2502.01922*, 2025.

Sungwon Han, Jinsung Yoon, Sercan O. Arik, and Tomas Pfister. Large language models can automatically engineer features for few-shot tabular learning. In *Proceedings of the 41st International Conference on Machine Learning (ICML 2024)*, 2024. arXiv:2404.09491.

Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. Tabllm: Few-shot classification of tabular data with large language models. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS 2023)*, volume 206 of *Proceedings of Machine Learning Research*, pp. 5549–5581, 2023. arXiv:2210.10723.

Noah Hollmann, Samuel Müller, and Frank Hutter. Large language models for automated data science: Introducing caafe for context-aware automated feature engineering. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*, 2023. arXiv:2305.03403.

Sirui Hong, Yizhang Lin, Bang Liu, Bangbang Liu, Binhao Wu, Ceyao Zhang, Chenxing Wei, Danyang Li, Jiaqi Chen, Jiayi Zhang, Jinlin Wang, Li Zhang, Lingyao Zhang, Min Yang, Mingchen Zhuge, Taicheng Guo, Tuo Zhou, Wei Tao, Xiangru Tang, Xiangtao Lu, Xiawu Zheng, Xinbing Liang, Yaying Fei, Yuheng Cheng, Zhibin Gou, Zongze Xu, and Chenglin Wu. Data interpreter: An llm agent for data science, 2024. arXiv:2402.18679.

Cheng-Yu Hsieh et al. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. *arXiv preprint arXiv:2305.02301*, 2023.

Congcong Hu, Yuang Shi, Fan Huang, Yang Xiang, Zhou Ye, Ming Jin, and Shiyu Wang. Eventcast: Hybrid demand forecasting in e-commerce with llm-based event knowledge, 2026. arXiv:2602.07695.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022. arXiv:2106.09685.

Qian Huang et al. Mlagentbench: Evaluating language agents on machine learning experimentation. *arXiv preprint arXiv:2310.03302*, 2023.

Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixing Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. Aide: Ai-driven exploration in the space of code, 2025. arXiv:2502.13138.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-llm: Time series forecasting by reprogramming large language models. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.01728.

Liqiang Jing et al. Dsbench: How far are data science agents from becoming data science experts? *arXiv preprint arXiv:2409.07703*, 2024.

Andrew Kiruluta. Filtering beats fine tuning: A bayesian kalman view of in context learning in llms. *arXiv preprint arXiv:2601.06100*, 2026.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*, 2022.

Patrick Tser Jern Kon et al. Curie: Toward rigorous and automated scientific experimentation with ai agents. *arXiv preprint arXiv:2502.16069*, 2025.

Quyu Kong et al. Language-tpp: Integrating temporal point processes with language models. *arXiv preprint arXiv:2502.07139*, 2026.

Suhas Kotha, Jacob Mitchell Springer, and Aditi Raghunathan. Understanding catastrophic forgetting in language models via implicit inference. *arXiv preprint arXiv:2309.10105*, 2024.

Jaris Küken, Lennart Purucker, and Frank Hutter. Large language models engineer too many simple features for tabular data. *arXiv preprint arXiv:2410.17787*, 2024.

Andrew K. Lampinen et al. On the generalization of language models from in-context learning and finetuning: a controlled study. *arXiv preprint arXiv:2505.00661*, 2025.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3045–3059, 2021. arXiv:2104.08691.

Nian Li, Chen Gao, Mingyu Li, Yong Li, and Qingmin Liao. Econagent: Large language model-empowered agents for simulating macroeconomic activities. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL 2024)*, 2024a. arXiv:2310.10436.

Peibo Li, Maarten de Rijke, Hao Xue, Shuang Ao, Yang Song, and Flora D. Salim. Large language models for next point-of-interest recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 1463–1472, 2024b. arXiv:2404.17591.

Ziming Li, Qianbo Zang, David Ma, Jiawei Guo, Tuney Zheng, Minghao Liu, Xinyao Niu, Yue Wang, Jian Yang, Jiaheng Liu, Wanjun Zhong, Wangchunshu Zhou, Wenhao Huang, and Ge Zhang. Autokaggle: A multi-agent framework for autonomous data science competitions, 2024c. arXiv:2410.20424.

Yuebing Liang, Yichao Liu, Xiaohan Wang, and Zhan Zhao. Exploring large language models for human mobility prediction under public events. *Computers, Environment and Urban Systems*, 2024. arXiv:2311.17351.

Yuhua Liao, Zetian Wang, Qiangqiang Nie, and Zhenhua Zhang. Bridging the last mile of time series forecasting with llm agents, 2026. arXiv:2606.02497.

Haoxin Liu, Zhiyuan Zhao, Shiyu Li, and B. Aditya Prakash. Evaluating system 1 vs. 2 reasoning approaches for zero-shot time series forecasting: A benchmark and insights. *arXiv preprint arXiv:2503.01895*, 2025a.

Haoxin Liu et al. Time-mmd: Multi-domain multimodal dataset for time series analysis. *arXiv preprint arXiv:2406.08627*, 2024.

Zefang Liu and Yinzhu Quan. Tpp-llm: Modeling temporal point processes by efficiently fine-tuning large language models. *arXiv preprint arXiv:2410.02062*, 2024.

Zexi Liu et al. Ml-agent: Reinforcing llm agents for autonomous machine learning engineering. *arXiv preprint arXiv:2505.23723*, 2025b.

Alejandro Lopez-Lira and Yuehua Tang. Can chatgpt forecast stock price movements? return predictability and large language models, 2023. arXiv:2304.07619; also available on SSRN.

Hariharan Manikandan, Yiding Jiang, and J. Zico Kolter. Language models are weak learners. *arXiv preprint arXiv:2306.14101*, 2023.

Nazanin Mehrasa, Akash Abdu Jyothi, Thibaut Durand, Jiawei He, Leonid Sigal, and Greg Mori. A variational auto-encoder model for stochastic point processes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3165–3174, 2019.

Manuel Mondal, Ljiljana Dolamic, Gérôme Bovet, Philippe Cudré-Mauroux, and Julien Audiffren. Do large language models exhibit cognitive dissonance? studying the difference between revealed beliefs and stated answers. *arXiv preprint arXiv:2406.14986*, 2024.

Yixin Ou et al. Automind: Adaptive knowledgeable agent for automated data science. *arXiv preprint arXiv:2506.10974*, 2025.

Oded Ovadia et al. Fine-tuning or retrieval? comparing knowledge injection in llms. *arXiv preprint arXiv:2312.05934*, 2024.

Xin Qiu, Junlong Tong, Yirong Sun, Yunpu Ma, Wei Zhang, and Xiaoyu Shen. Rethinking the role of llms in time series forecasting. *arXiv preprint arXiv:2602.14744*, 2026.

Michael A Riegler, Kristoffer H Hellton, Vajira Thambawita, and Hugo L Hammer. Using large language models to suggest informative prior distributions in bayesian regression analysis. *Scientific Reports*, 15(1):33386, 2025.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

Xiaoming Shi, Siqiao Xue, Kangrui Wang, Fan Zhou, James Y. Zhang, Jun Zhou, Chenhao Tan, and Hongyuan Mei. Language models can improve event prediction by few-shot abductive reasoning. *arXiv preprint arXiv:2305.16646*, 2023.

Mingtian Tan, Mike A. Merrill, Vinayak Gupta, Tim Althoff, and Thomas Hartvigsen. Are language models actually useful for time series forecasting? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. Spotlight. arXiv:2406.16964.

Alicia Y. Tsai et al. Leveraging llm reasoning enhances personalized recommender systems. *arXiv preprint arXiv:2408.00802*, 2024.

Benjamin Turtel et al. Training large language models to predict clinical events. *arXiv preprint arXiv:2605.12817*, 2026.

Mahsa Valizadeh, Xiangjue Dong, Rui Tuo, and James Caverlee. Language models as semantic augmenters for sequential recommenders. *arXiv preprint arXiv:2510.18046*, 2025.

Dongsheng Wang, Yuxi Huang, Shen Gao, Yifan Wang, Chengrui Huang, and Shuo Shang. Generative next POI recommendation with semantic ID. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2025a. doi: 10.1145/3711896.3736981. arXiv:2506.01375.

Ruocheng Wang, Eric Zelikman, Gabriel Poesia, Yewen Pu, Nick Haber, and Noah Goodman. Hypothesis search: Inductive reasoning with language models. In *International Conference on Learning Representations*, volume 2024, pp. 38993–39014, 2024a.

Xinyi Wang et al. Large language models are latent variable models: Explaining and finding good demonstrations for in-context learning. *arXiv preprint arXiv:2301.11916*, 2023.

Yihan Wang, Andrew Bai, Nanyun Peng, and Cho-Jui Hsieh. On the loss of context-awareness in general instruction fine-tuning. *arXiv preprint arXiv:2411.02688*, 2024b.

Yihan Wang et al. Two-stage llm fine-tuning with less specialization and more generalization. *arXiv preprint arXiv:2211.00635*, 2022.

Ziwei Wang, Jie Zhou, Qin Chen, Min Zhang, Bo Jiang, Aimin Zhou, Qinchun Bai, and Liang He. Llm-kt: Aligning large language models with knowledge tracing using a plug-and-play instruction, 2025b. arXiv:2502.02945.

Hans Weytjens and Ingo Weber. David vs. goliath in next activity prediction: Argmax vs. lstm, transformer, and llm. *arXiv preprint arXiv:2606.15868*, 2026.

Andrew Robert Williams et al. Context is key: A benchmark for forecasting with essential textual information. *arXiv preprint arXiv:2410.18959*, 2025.

Qingyun Wu et al. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.

Zijun Wu, Zhitian Zhang, Seyed Hossein Hajimirsadeghi, Nikita Dvornik, and Alexander Pashevich. Can LLMs access their world knowledge for event prediction? In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2026.

Yunjia Xi, Weiwen Liu, Jianghao Lin, Xiaoling Cai, Hong Zhu, Jieming Zhu, Bo Chen, Ruiming Tang, Weinan Zhang, Rui Zhang, and Yong Yu. Towards open-world recommendation with knowledge augmentation from large language models. In *Proceedings of the 18th ACM Conference on Recommender Systems (RecSys 2024)*, 2024. arXiv:2306.10933.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*, 2022.

Hao Xue and Flora D. Salim. Promptcast: A new prompt-based learning paradigm for time series forecasting. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2023. arXiv:2210.08964.

Khurram Yamin, Gaurav Ghosal, and Bryan Wilder. Can llms reconcile knowledge conflicts in counterfactual reasoning? *arXiv preprint arXiv:2506.15732*, 2025.

Xiaodong Yan, Tengwei Song, Yifeng Jiao, Jianshan He, Jiaotuan Wang, Ruopeng Li, and Wei Chu. Spatio-temporal hypergraph learning for next POI recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 403–412, 2023. doi: 10.1145/3539618.3591770.

Song Yang, Jiamou Liu, and Kaiqi Zhao. GETNext: Trajectory flow map enhanced transformer for next POI recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 1144–1153, 2022. doi: 10.1145/3477495.3531983.

Penglong Zhai, Jie Li, Fanyi Di, Yue Liu, Yifang Yuan, Jie Huang, Peng Wu, Sicong Wang, Mingyang Yin, Tingting Hu, Yao Xu, and Xin Li. Cognitive-aligned spatio-temporal large language models for next point-of-interest prediction. *arXiv preprint arXiv:2510.14702*, 2025.

Minda Zhao, Yilun Du, and Mengyu Wang. Large language models are bad dice players: Llms struggle to generate random numbers from statistical distributions. *arXiv preprint arXiv:2601.05414*, 2026.

Yushang Zhao, Xinyue Han, Qian Leng, Qianyi Sun, Haotian Lyu, and Chengrui Zhou. Efficient cold-start recommendation via bpe token-level embedding initialization with llm, 2025. arXiv:2509.13179.

Yizhen Zheng, Huan Yee Koh, Jiaxin Ju, Anh T. N. Nguyen, Lauren T. May, Geoffrey I. Webb, and Shirui Pan. Large language models for scientific discovery in molecular property prediction. *Nature Machine Intelligence*, 7, 2025. doi: 10.1038/s42256-025-00994-z.

Tian Zhou, Peisong Niu, Xue Wang, Liang Sun, and Rong Jin. One fits all: Power general time series analysis by pre-trained lm. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. Spotlight. arXiv:2302.11939.

Yinghao Zhu, Zixiang Wang, Junyi Gao, Yuning Tong, Jingkun An, Weibin Liao, Ewen M. Harrison, Liantao Ma, and Chengwei Pan. Prompting large language models for zero-shot clinical prediction with structured longitudinal electronic health record data, 2024. arXiv:2402.01713.

Simiao Zuo, Haoming Jiang, Zichong Li, Tuo Zhao, and Hongyuan Zha. Transformer hawkes process. In *International conference on machine learning*, pp. 11692–11702. PMLR, 2020.

A Inference Time Prior Querying Hypothesis

For many prediction tasks in the real world, the data generation process is a result of an extremely complex system that is often driven by the aggregation of many various human decisions. For example, forecasting product demand depends on consumer decisions which depend on many unknown factors we cannot possibly hope to capture in their entirety in our model. The Inference Time Prior Querying (ITPQ) Hypothesis states that any static model (i.e. a model unable to reason) necessarily compresses information about the underlying data generating system which is lossy. This suggests that the only way to maximize signal contained in our understanding of the data generating mechanism is via a model that is able to reason at inference time since it can query its uncompressed world model for each circumstance. Traditional statistical modeling assumes that data is sampled from a static distribution and thus with enough observations a correctly specified model can become arbitrarily precise (minimize risk). One way to frame the ITPQ Hypothesis is through the lens that most data generating mechanisms in the real world do not produce IID samples from a static distribution and often in practice, one will observe prediction scenarios which are Out of Distribution (OOD) relative to the training set. Further, what is considered OOD depends on the degree to which one expects patterns and the model capturing them to generalize. In the absence of data, reasoning about our understanding of the data generating mechanism may be the best path forward (in contrast to blindly trusting a models extrapolation to new territory). It has been demonstrated for time series that a benefit of LLMs is in terms of cross domain generalization (Qiu et al., 2026). The argument for the value of reasoning at inference time only becomes stronger as data becomes more scarce since a greater fraction of future observations 'appear' OOD (or sufficiently different from the observed data). Another lens to view the ITPQ Hypothesis is that of information compression. Taking a human subject matter expert as a prediction model, one could ask the expert to write down everything they know on the topic of interest and then afterwards when a question arises, look at the document for the answer. It is quite possible the answer may not be in the document but if asked directly to the human expert they could provide an answer. The set of questions one can answer is likely far greater than the set they can anticipate. This is especially true when the information attempted to be distilled pertains to a system so complex it could not possibly be written down in it's entirety. Thus preserving the world model in it's entirety and querying it as needed is the only way to avoid loss of information.

B Related Work (Extended)

LLMs as carriers of a predictive world prior. A growing body of work reports that large language models supply genuine predictive signal on quantitative tasks *because* their pretraining encodes a semantic model of the world, rather than because they are flexible sequence learners. The clearest demonstrations appear where the predictive target depends on knowledge the numerical data alone does not contain. In finance, Lopez-Lira & Tang (2023) show that GPT-4 anticipates market reactions from news headlines released *after* its training cutoff, with the ability emerging only at scale and framed as an emergent financial-reasoning capacity; LLM-derived news representations likewise improve portfolio returns beyond conventional sentiment scores (Guo & Hauptmann, 2024).[1] In recommendation, Xi et al. (2024) attribute open-world gains to the LLM's reasoning over user preferences and factual item knowledge, and LLM token embeddings act as semantic priors enabling cold-start prediction with no interaction history (Zhao et al., 2025). The pattern recurs far from time series and tabular data: few-shot prompting of structured electronic health records rivals or beats fully supervised models (Cui et al., 2024; Zhu et al., 2024) (though only in the data-scarce regime, with conventional models regaining the lead once labels are plentiful); an LLM's scientific knowledge drives an interpretable predictor to state-of-the-art molecular property prediction (Zheng et al., 2025); human-mobility models exploit reasoning over event text and geographic knowledge (Liang et al., 2024; Feng et al., 2025); and epidemic forecasting is recast as text reasoning over policies and surveillance (Du et al., 2025). Beyond prediction

[1]The strongest reported figures concern the *non-tradable* initial reaction; we cite these as evidence of encoded signal, not of tradable profit.

proper, LLMs reproduce human survey distributions (Argyle et al., 2023), model student knowledge (Wang et al., 2025b), forecast event-driven demand (Hu et al., 2026), estimate consumer willingness-to-pay (Brand et al., 2023), and simulate macroeconomic dynamics (Li et al., 2024a). A useful contrast is Gong et al. (2025), which deploys a *frozen* LLM as a generic next-token predictor over purely numerical series with no textual knowledge, isolating the "LLM-as-sequence-model" usage from the "LLM-as-world-prior" usage we study.

Three set-ups for deploying the prior: direct, distilled, and hybrid. We organize prior work by *how* the LLM's prior is brought to bear at prediction time, which maps onto our three set-ups. In the **direct** set-up (S1) the LLM *is* the predictor at inference. For time series this means serializing numeric values into text for a frozen model (Gruver et al., 2023; Xue & Salim, 2023) or adapting the backbone by reprogramming and lightweight tuning (Jin et al., 2024; Zhou et al., 2023; Cao et al., 2024; Ansari et al., 2024). The same "LLM-as-model" pattern recurs for tabular data (Hegselmann et al., 2023), recommendation (Xi et al., 2024; Zhao et al., 2025; Wang et al., 2025b), finance (Lopez-Lira & Tang, 2023; Guo & Hauptmann, 2024), clinical and epidemic forecasting (Cui et al., 2024; Zhu et al., 2024; Du et al., 2025; Gong et al., 2025), mobility (Feng et al., 2025), and social or economic simulation (Li et al., 2024a; Argyle et al., 2023).

In the **distilled** set-up (S2) the LLM is used *once* to build a lightweight static model and is then removed from the inference path, precisely the "LLM as an automated data scientist" setting, in which the LLM proposes features, rules, or pipelines that a conventional model then executes with no per-instance LLM call. Instances range from feature synthesis (Hollmann et al., 2023; Han et al., 2024; Abhyankar et al., 2025) and text-serialised few-shot classification (Hegselmann et al., 2023) to end-to-end agents that automate the workflow (Hong et al., 2024; Jiang et al., 2025; Guo et al., 2024; Li et al., 2024c; Chi et al., 2024; Ou et al., 2025), some reaching expert-competition performance (Grosnit et al., 2024) or adding experimental rigour (Kon et al., 2025) and reinforcement-learned policies (Liu et al., 2025b); earlier work framed the build step as hyperparameter search (Chen et al., 2022). Progress is tracked by ML-engineering benchmarks (Chan et al., 2025; Huang et al., 2023; Jing et al., 2024) and surveyed by Chen et al. (2025) and Fang et al. (2024); the pattern is not confined to tabular data, appearing in science (Zheng et al., 2025) and recommendation (Xi et al., 2024). Conceptually closest to S2 is Wang et al. (2024a), whose model proposes candidate rules as natural-language hypotheses, implements each as an executable Python program, and keeps those that best fit the observed data. In contrast, S2 fits a single deployable predictor to a specific TPP training corpus rather than searching over programs to explain a handful of examples that themselves are generated based on a set of simple rules. Our S2 is a deliberately constrained instance: rather than maximizing leaderboard accuracy, the agent must distill its *world knowledge* into a small, LLM-free `predict.py`. This distinguishes it from the concurrent EVIL (Berghaus, 2026), which also evolves an LLM-free program for event prediction but as a single zero-shot script applied identically to every dataset — a program-search engine, not a per-domain world-model distillation.

In the **hybrid** set-up (S3) the LLM reasons at inference *with* the model it built in S2 as a callable tool, building on CoT prompting (Kojima et al., 2022) and tool/program augmented LLMs (Schick et al., 2023; Gao et al., 2023; Wu et al., 2023) which outsource computation, motivated by documented LLM failures at exact arithmetic and sampling (Zhao et al., 2026; Mondal et al., 2024). The closest prior configurations expose a trained model as a tool an LLM agent calls (Chen & Bibi, 2026) or have an LLM revise a trained forecaster's output using retrieved context (Liao et al., 2026); neither, however, treats the configuration as a regime to be *studied*. Our contribution is a controlled, head-to-head study of all three set-ups across domains, isolating the contribution of inference-time reasoning and mapping where tool use helps (computation-dominated tasks) versus where unaided reasoning already wins (knowledge-dominated tasks).

LLMs for temporal point processes and event sequences. Classical neural TPP models parameterise the conditional intensity with recurrent or attention mechanisms (Zuo et al., 2020; Mehrasa et al., 2019). More recently LLMs have been applied to asynchronous event sequences, almost always *as* the predictor: Gupta et al. (2025) finetune an LLM for asynchronous time series (the direct antecedent of our finetuning arm), Liu & Quan (2024) and Kong et al. (2026) finetune LLMs to model event types and times, Chen et al. (2026) sharpen their temporal awareness, Shi et al. (2023) use few-shot abductive reasoning to improve an event model, and Hu et al. (2026) forecast event sequences. We adopt the DESCRIBEEVENT benchmark (Wu et al., 2026), which pairs symbolic event types with natural-language descriptions; our S1 re-implements its strongest zero-shot configuration. None of this work contrasts a *finetuned* predictor with a *reasoning* one on the same task. Consistent with our finding that finetuning wins where the pattern is learnable, Weytjens & Weber (2026) report that on learnable event logs a trivial count-based baseline already matches billion-parameter LLMs.

Finetuning versus in-context reasoning, and out-of-distribution robustness. A central comparison of our experiments is between finetuning and in context reasoning as a means for prediction. Our experiments join two literatures that, to our knowledge, have not been brought together. The first shows that **supervised finetuning narrows an LLM's use of in-context information**: finetuning on a single task degrades general in-context learning through an early *format specialisation* (Wang et al., 2022); instruction tuning induces an attention bias away from user-provided context toward template tokens, so tuned models rely *less* on the prompt when parametric knowledge suffices (Wang et al., 2024b); and finetuning skews the model's implicit task inference toward the finetuning distribution, suppressing rather than erasing prompt-recoverable capabilities (Kotha et al., 2024). In matched-data comparisons the same information generalises more flexibly in context than when instilled by finetuning (Lampinen et al., 2025), a contrast echoed by "SFT memorises, RL generalises" (Chu et al., 2025) and by knowledge-injection studies where in-context or retrieval presentation beats finetuning for new facts (Ovadia et al., 2024; Gekhman et al., 2024; Yamin et al., 2025). Crucially, this literature is almost entirely situated in natural-language and question-answering settings; whether the same trade-off governs *quantitative* prediction, where the finetuned model has learned a genuine data-generating pattern rather than a surface format, has not been tested. The second literature shows that **prompted LLMs are valuable precisely out-of-distribution**: they are argued to add value through cross-domain and OOD generalisation rather than in-distribution fit (Qiu et al., 2026; Valizadeh et al., 2025); on a benchmark where context is necessary by design, prompting a large model to read it beats strong statistical and time-series-foundation baselines while smaller models fail, and instruction tuning degrades the raw numerical-completion mode even as it enables the context-using one (Williams et al., 2025); and Ashok et al. (2025) name the residual failure an *execution gap*; models correctly explain how context should change a forecast yet fail to apply it. With further benchmarks corroborating that reasoning pays off exactly when exploitable context is present (Liu et al., 2025a; 2024). Crucially, these studies evaluate *only prompted models* and attribute the residual failure to model *scale*; none contrasts a finetuned predictor against a reasoning model on the same task. The closest prior comparison, Tsai et al. (2024), studies LLM reasoning for recommendation in both prompted and finetuned settings, but frames reasoning as an accuracy enhancement rather than as a test of which regime can absorb new information under a distribution shift. A theoretical thread anticipates the joint outcome: in-context learning is implicit Bayesian inference over a latent concept that indexes the data-generating process (Xie et al., 2022; Wang et al., 2023), in-context and in-weight solutions are distinct and can trade off (Chan et al., 2022) with the in-context route diminishing under further training (Chan et al., 2024), and inference-time adaptation can be cast as online Bayesian filtering of which gradient finetuning is a degenerate, noise-free limit (Kiruluta, 2026), predicting exactly that a finetuned predictor forfeits inference-time adaptability. Our distribution-shift experiment provides the missing intersection of these lines in a single controlled setting: (i) a *quantitative* next-event task rather than text or QA; (ii) an actual finetuned-small-model-versus-reasoning-model head-to-head, and (iii) a controlled regime shift: the finetuned model is trained on one part of the distribution and tested on another, while the reasoning model, having no training distribution to diverge from, is invariant by construction. Thus its out-of-distribution robustness is a clean baseline rather than a model-quality confound. The result localizes the OOD failure to *finetuning itself*, not to model *scale* (the axis the context-use literature varied) and not to model *class*, LLM versus traditional model (the axis the OOD-value literature varied). We report the boundary honestly: where inputs are structured and labels plentiful, finetuning or distillation can match or beat a prompted reasoner (Turtel et al., 2026; Hegselmann et al., 2023), and the low-data "LLM edge" on tabular tasks narrows as data grows (Manikandan et al., 2023).

C How LLMs can distill their world knowledge into an external model

In this section we discuss the various ways in which an LLM acting as a model builder can distill their world knowledge into an external model. We highlight three main ways in which this world knowledge can seep into the final model. The first is through the hypothesis class. This includes the type of model and architecture and is an opportunity to align inductive bias in the chosen model with the data / domain. The second is through feature engineering. Taking the data in its most general form with maximal information (such as what each predictor/variable corresponds to and the various relationships among these variables) and refining it in a way that aligns with the chosen architecture and is minimally lossy. Lastly, there is incorporation of prior knowledge about the distribution which is being predicted over. For example, understanding hard constraints over the distribution or existing knowledge over the weighting of various outcomes. This can be implemented via model constraints or through various heuristics. In our experiments, we see the first two methods come into play significantly but the last remains mostly absent. We suspect that the difficulty of incorporating general distributional knowledge into a models architecture or weights is limiting, along with the difficulty in anticipating all of the heuristics which may be valuable (aligning with the Inference Time Querying Hypothesis).

D Dataset Size Ablation

	S1	S2 accuracy at training-data fraction				
Dataset	(prior)	100%	25%	10%	5%	2.5%
GitHub Repo	0.535	0.512±0.014	0.461±0.037	0.443±0.038	0.420±0.016	0.414±0.010
GitHub User	0.495	0.471±0.005	0.435±0.012	0.433±0.003	0.419±0.010	0.386±0.025
Earthquake	0.382	0.384±0.012	0.355±0.031	0.363±0.034	0.340±0.024	0.360±0.026
GDELT	0.222	0.227±0.019	0.223±0.018	0.223±0.029	0.210±0.044	0.209±0.023
NBA	0.582	0.594±0.006	0.569±0.018	0.578±0.015	0.591±0.008	0.550±0.031
Wikipedia	0.389	0.393±0.031	0.373±0.017	0.355±0.010	0.353±0.020	0.335±0.039
Amazon	0.250	0.271±0.009	0.269±0.007	0.256±0.015	0.252±0.006	0.253±0.004

Table 3: **Restricted-data S2 ablation.** S2 (computational prior) accuracy as the pre-amble's training data is cut to 25/10/5/2.5% of full; S1 (verbal prior) is data-invariant and shown as a reference. S2 is strikingly data-efficient, even at 2.5% (a $40\times$ cut) its accuracy stays within $\sim$0.02–0.10 of the full-data value on every dataset. We note that this pattern is not as strong when measuring Macro-F1, suggesting that class imbalance may skew our perception of the accuracy metric. All fractions are mean±std over 3 independent subsample seeds.

E CoT conditioned FineTune results

	S3-Opus			FT			FT CoT		
Domain	Acc	F1	MAE	Acc	F1	MAE	Acc	F1	MAE
NBA	.607	**.405**	.062	**.629**	.399	**.060**	.626†	.385	.063
Wikipedia	.382	.408	**16.36**	**.414**	**.432**	16.59	.411	.422	16.55
Earthquake	.390	**.255**	**.326**	**.404**	.199	.431	.401†	.177	.431
GDELT	.220	**.157**	**9.61**	**.242**	.078	12.14	.241†	.100	12.14
Amazon	**.283**	**.172**	**5.67**	.276	.154	5.76	.260†	.154	5.76
GitHub Repo	.548	.491	**2.11**	.569	.500	2.15	**.574**	**.506**	2.15
GitHub User	.502	.428	**1.41**	**.533**	**.459**	1.42	.365	.287	1.43

Table 4: Finetuned Qwen3.5-9B with and without Chain of Thought conditioning, referenced against S3 from the previous table. FT cells are the validation-selected model from 3 independently trained seeds. †: single training run trained on a 2000 data point CoT-conditioned subset of the full train set.

F Latency Accuracy Plot

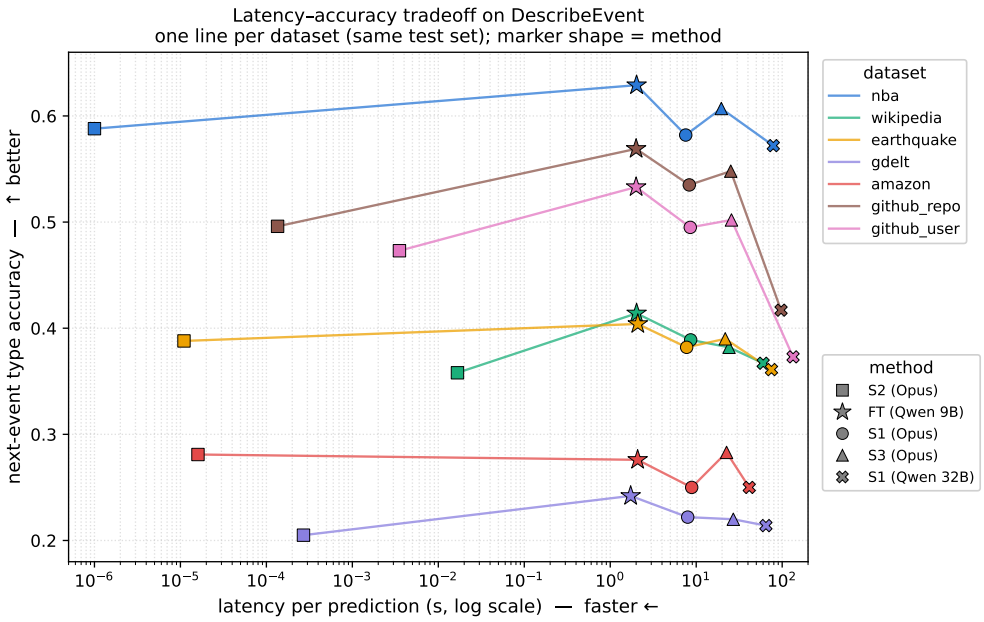


Figure 4: A visual comparison of accuracy and latency across many of the tested approaches. The latency is in seconds and measures average inference time for a single prediction. Build or train time for set ups is not considered here. Important for comparison is hardware used: All opus approaches are run via a Claude Code plan (hardware not disclosed). All QWEN approaches are run on a single A100 40GB GPU. S2 Opus (and the identical model used by Opus in S3) are run on cpu.

G Data Scale

Another important consideration when designing LLM powered prediction system is the quantity of data that is accessible. The literature on LLMs for TPP and other domains such as tabular data, as well as our own work, has shown that LLMs offer strong performance particularly in the limited data regime (Hegselmann et al., 2023; Manikandan et al., 2023; Cui et al., 2024). In contrast using an LLM to build a model as in S2 requires access to enough data to fit a model. We thus expect the performance of S2 to deteriorate as the available data shrinks towards the zero shot domain that S1 operates in. We thus conduct an experiment to artificially restrict the training data that S2 has access to, in order to observe the role of training data magnitude on S2 performance. We randomly sample a subset of training samples of 25%, 10%, 5% and 2.5% with 2.5% resulting in ≤ 10 sequences for each dataset. Results can be found in table 5. The general trend is as expected, performance decays with dataset size. The rate of decay in performance however is strongly varying by dataset. The same three datasets which we suspect have the strongest reliance on prior world knowledge (wikipedia, github repo and github user) have the steepest decline, whereas the other four datasets are surprisingly robust). We emphasize that the lower tail of 5-10% samples is small enough that manually collecting data at this scale is feasible for most applications, which points towards the S2 approach as being accessible in many settings. These results also give some insight into the variance (across only 3 seeds) of the S2 approach. The numbers reported in this table are different from those in table 1 where we consider just 1 random seed. We can see that the performance is subject to non-negligible variance.

H Prompting Setup

S1 and S3 receive the event history in their prompts as text, using Wu et al. (2026)'s "Symbolic + Description" serialization (real type names anonymised to `type_0`,...,`type_K`; and, except on NBA, the real name prepended to that event's description); S3 is exactly the S1 prompt plus an appended block that exposes a tool. S2 is different: it never sees that per-step serialization at all. It runs once as an agentic coding session that is handed the *raw training data as*

Dataset	S1 (prior)	S2 macro-F1 at training-data fraction				
		100%	25%	10%	5%	2.5%
GitHub Repo	0.462	0.415±0.012	0.351±0.016	0.325±0.009	0.297±0.023	0.271±0.014
GitHub User	0.415	0.347±0.038	0.302±0.015	0.282±0.023	0.263±0.006	0.229±0.028
Earthquake	0.253	0.237±0.018	0.234±0.024	0.191±0.057	0.216±0.054	0.157±0.084
GDELT	0.141	0.127±0.046	0.142±0.013	0.123±0.060	0.111±0.012	0.137±0.012
NBA	0.361	0.404±0.012	0.398±0.021	0.396±0.028	0.400±0.014	0.341±0.047
Wikipedia	0.399	0.366±0.039	0.306±0.020	0.295±0.047	0.221±0.064	0.237±0.013
Amazon	0.160	0.175±0.006	0.146±0.026	0.181±0.012	0.184±0.007	0.173±0.020

Table 5: **Restricted-data S2 ablation (macro-F1).** Companion to Table 3 with type macro-F1 in place of accuracy. S1 (verbal prior) is data-invariant. All fractions are mean±std over 3 independent subsample seeds.

a JSONL file in its workspace and must build a stand-alone predictor; the event descriptions reach it only as fields of that JSON. We give S1, S2, and S3 in turn, then the per-dataset descriptions that drop in to these templates (§H.4).

Notation. In every prompt listing below, a token in **double angle brackets** — e.g. `<<DOMAIN>>` or `<<TIME_KEY>>` — is a value the harness fills in before the prompt is sent to the model (per domain, or per step). *Everything else is literal prompt text, reproduced verbatim.* In particular, single-angle slots such as `<type>` and `<positive_number>` are *part of* the prompt — they are format instructions the *model* is asked to fill in, not substitutions we make — and JSON braces {...} are literal. A leading "↪" marks a line that is merely wrapped for the page, not a new line.

H.1 S1 — VERBAL PRIOR (REASON AT INFERENCE)

At each test step the model gets one independent, teacher-forced call (no memory across steps, no tools). The call is built from two pieces, a fixed *system block* and a per-step *user block* (the serialized history), which are concatenated into a single string with the system block wrapped in `<system>...</system>` tags (the exact assembly is shown at the end of this subsection; the open-model path instead sends the two blocks as the `system` and `user` chat roles).

System block. `<<VALID_EVENT_TYPES>>` is the sequence's anonymised vocabulary (`type_0, ..., type_K`); `<<UNIT>>` is the one-letter time suffix (e.g. `h`); `<<TIME_KEY>>` is the time field name (e.g. `time_hours`).

```
You predict the next event in a sequence. Given the event history, predict the next event type and time.

Valid event types: <<VALID_EVENT_TYPES>>.

Each event shows: [type] +X<<UNIT>> (<<TIME_KEY>> since previous event) | description, where description
    ↪ provides details about the event (may be truncated). Use the event types, their descriptions, and
    ↪ the +X<<UNIT>> time gaps to predict what happens next and when.

Response format (follow exactly):
Brief reasoning here (2-3 sentences max).
{"event_type": "<type>", "<<TIME_KEY>>": <positive_number>}
```

User block (the serialized history). The history is a numbered, chronological list. Each event occupies one line of the form

`INDEX. [TYPE] +DT UNIT | NAME: DESCRIPTION`

where `INDEX` is the 1-based position; `TYPE` is the anonymised label `type_K`; `DT` is the inter-arrival time since the previous event, printed to three decimals, immediately followed by the domain's `UNIT` suffix; `NAME` is the event type's real name, prepended *only* when the domain sets `prepend_semantic_label` (it is hidden on NBA, forcing symbolic-only reasoning); and `DESCRIPTION` is the event's free text, truncated to 500 characters. An event with no description drops the `| NAME: DESCRIPTION` part entirely. A closing instruction line follows the list. A concrete example (Wikipedia; `type_5` = revert, `type_0` = content_addition):

```
Event history (chronological):
1. [type_5] +0.000h | revert: Reverted edits by 192.0.2.5 to last version by Alice
```

```
2. [type_0] +2.417h | content_addition: Added a paragraph on the 1998 final ...
   (one line per event, oldest to newest)

Predict the next event type and <<TIME_KEY>>.
```

Assembled prompt. The two blocks are sent as a single string of this shape (the open-model path uses the same two blocks as `system`/`user` chat messages instead):

```
<system>
[System block, above]
</system>

[User block, above]
```

The reply is parsed for a trailing JSON object; the free text before it is the chain-of-thought (examples in §J). Only `type_K` labels and a positive number are accepted, and `time == 0` is a valid prediction.

The anonymised-label "Symbolic + Description" prompting style follows the DESCRIBEEVENT benchmark (Wu et al., 2026): an event's true class name is written at the start of its description, so the label's meaning remains available to the model despite the `type_K` anonymisation (on NBA, whose play-by-play descriptions already name the event in full, the labels are left purely symbolic).

H.2 S2 — COMPUTATIONAL PRIOR (BUILD THE MODEL ONCE)

S2 runs *once* per domain as an agentic Claude-CLI session with full bash and file access in a private workspace. Unlike S1/S3 it is not shown a serialized history; instead the raw training histories are symlinked into its workspace as a JSONL file (one sequence per line, with `type_event`, `time_since_last_event`, and `type_text` arrays), and the descriptions reach the model only as fields of that JSON. The session does EDA, builds and validates candidate models on held-out splits of that data, and writes a single `predict.py` that the evaluation harness then calls for *every* test prediction — with **no LLM in the loop at inference**. The full pre-amble (`build_pretrain_prompt`) is reproduced verbatim below. Following the notation above, `<<DOMAIN>>` is the dataset name, `<<DOMAIN_DESCRIPTION>>` is the drop-in domain blurb (§H.4), `<<VALID_EVENT_TYPES>>` the vocabulary block, `<<TIME_KEY>>` / `<<TIME_UNIT>>` the time field and unit, and `<<CATEGORY_FIELD>>` an extra per-category line present only for Amazon.

Pre-amble (verbatim).

```
You are a world-class machine learning researcher specializing in temporal point processes (TPP).
Your task: build the best possible model for next-event prediction on <<DOMAIN>> sequences,
then write predict.py --- a script the evaluation harness will call for every test prediction.

------------------------------------------------------------
WORKING STYLE (IMPORTANT --- you have a LIMITED context window)
------------------------------------------------------------

Your context window is small relative to this task, so work in a FILE-CENTRIC way
and keep what you read into the conversation minimal --- otherwise you will run out
of room and forget your earlier work:
  - Put logic in .py scripts and RUN them; do not paste large code or data inline.
  - NEVER cat/print large files or full datasets to the terminal. When inspecting
    data, print only small summaries (shapes, counts, a few rows) --- write detailed
    EDA results to files under notes/ instead of dumping them to stdout.
  - Maintain notes/progress.md as you go: after each major step, append a few lines
    (what you did, key numbers, the next step). Treat it as your durable memory so
    you can recover if context is compacted.
  - Prefer many small, verifiable steps over one giant script.

------------------------------------------------------------
TASK: NEXT-EVENT PREDICTION ON TEMPORAL POINT PROCESSES
------------------------------------------------------------

Given: a history of N events (type + inter-arrival time + text description)
Predict: next event type AND inter-arrival time

<<DOMAIN_DESCRIPTION>>

------------------------------------------------------------
DATA FORMAT
------------------------------------------------------------

Training data: training_data  (this is a symlink to the train_histories.jsonl file)
Each line is a JSON object (one sequence):
```

```
  {
    "seq_idx": 0,
<<CATEGORY_FIELD>>    "type_event": ["event_type_1", "event_type_2", ...],      # list of N strings
    "time_since_last_event": [0.0, 1.2, 3.4, ...],           # list of N floats (<<TIME_UNIT>>)
    "type_text": ["description 1", "description 2", ...],     # list of N strings (may be empty)
    "description": "sequence-level context",                   # string
    "history_len": N,                                          # number of events in this training history
    "original_seq_len": M                                      # total sequence length (for reference)
  }

<<VALID_EVENT_TYPES>>
Time key: <<TIME_KEY>> (inter-arrival times in <<TIME_UNIT>>)

------------------------------------------------------------
AVAILABLE TOOLS & RESOURCES
------------------------------------------------------------

You are running as a Claude CLI session with --dangerously-skip-permissions.
You have FULL bash access. You can:
  - Read/write files in this workspace directory
  - Install Python packages: pip install <package>
  - Run Python scripts: python script.py
  - Use PyTorch, numpy, scipy, pandas, scikit-learn (likely already installed); pip install anything else
  - If a GPU is available, PyTorch will detect it automatically (check with torch.cuda.is_available())

------------------------------------------------------------
PROCESS TO FOLLOW
------------------------------------------------------------

Step 1 --- EDA
  - Load training_data, understand data distributions
  - Event type frequencies, class imbalance
  - Inter-arrival time distributions (log-scale, by type)
  - Correlation between type_text descriptions and event types
  - Sequence-level statistics (length, rate, etc.)
  - Save EDA plots/stats to notes/eda_summary.txt

Step 2 --- Encode what matters into the model (the most important step)
  The type_text descriptions are the key signal distinguishing this from plain TPP data. The hard
  constraint is the S2 contract: at inference predict.py runs ALONE --- NO LLM. So any understanding you
  want the model to use --- including your own world knowledge about <<DOMAIN>> and what the descriptions
  MEAN --- must be encoded into the model's weights, features, or artifacts DURING THIS SESSION. Read the
  descriptions, work out for YOURSELF which signals matter (they are not given to you), and bake them in
  however you judge best. If you conclude some signal genuinely cannot be encoded without an LLM at
  inference, document that explicitly as a finding.
  Write notes/prior_knowledge.md: your prior knowledge about <<DOMAIN>>, which signals you think are
  predictive and why, and the encoding strategy you chose.

Step 3 --- Build and compare approaches
  Implement the approaches you chose (methods and architecture entirely your call), validate each
  honestly on held-out splits of the training data (see the generalization-discipline note below), and
  compare them on the same splits. Save per-approach validation results (type accuracy / macro-F1, time
  MAE) to notes/validation_results.txt.

Step 4 --- Select the final model
  Pick the best approach (or an ensemble) by honest held-out performance. A model that uses only event
  type + time --- ignoring the descriptions and the knowledge you can bring to them --- is leaving signal on
  the table; aim higher.

Step 5 --- Iterate
  - Debug issues, tune hyperparameters
  - Try different architectures if first attempt fails
  - Save best model weights to models/ directory

Step 6 --- Write predict.py
  (If some description signal genuinely cannot be encoded without an LLM at inference, document that
  explicitly in notes/approach.md --- it is a meaningful finding, not a failure. predict.py must NOT call
  an LLM.)

  CRITICAL: implement exactly this interface:

  python predict.py --history_json PATH [--verbose]

  Where:
    --history_json PATH: path to a JSON file containing a list of event dicts:
      [{"type_event": "revert", "time_since_last_event": 1.2, "type_text": "..."}, ...]
    --verbose: optional flag for debug output (to stderr only)

  Output to stdout (EXACTLY ONE LINE, no other stdout output):
    {"event_type": "<type>", "<<TIME_KEY>>": <positive_float>}
```

```
  The script must:
  - Load any saved model from models/ directory (relative to script location)
  - Accept history of any length (1 to hundreds of events)
  - Handle missing type_text gracefully (empty string ok)
  - Never crash on edge cases --- always output a valid JSON line
  - Be fast: ideally < 2 seconds per call (no model training at inference time)

  OPTIONAL (recommended --- big eval speedup): also support a batch mode
      python predict.py --batch_json PATH
    where PATH is a JSON LIST of histories (each history is itself a list of event
    dicts, same shape as --history_json). Load the model ONCE and emit one
    prediction per input history, EITHER as a single JSON array on one line:
      [{"event_type": "<type>", "<<TIME_KEY>>": <float>}, ...]
    OR as one JSON object per line, in the same order. This lets the harness avoid
    reloading the model thousands of times. If you implement it, the harness uses
    it automatically; if you don't, the harness falls back to per-call --history_json.

Step 7 --- Write notes/approach.md
  Document:
  - What approaches were tried
  - What worked and what didn't (with validation numbers)
  - Key insights about the data
  - How domain prior knowledge was (or wasn't) incorporated
  - Architecture of the final model
  - Any known limitations

Step 7.5 --- MANDATORY VERIFICATION (do NOT skip --- most failures happen here)
  Before signalling completion you MUST prove your work runs end-to-end. An
  automated gate will re-check this after you finish and bounce it back to you
  with the exact error if it fails --- so verify it yourself now:

  (a) Contract self-test. Run predict.py EXACTLY as the harness will, on a real
      history, and confirm the output:
        python predict.py --history_json <a real sample history>.json
      The ONLY thing on stdout must be one JSON line:
        {"event_type": "<type>", "<<TIME_KEY>>": <positive_float>}
      where <type> is one of the valid types above. Common failures to check:
        - extra prints on stdout (move ALL debug output to stderr)
        - event_type not in the valid set, or the wrong key name for time
        - predict.py can't find models/ (use a path relative to __file__)
        - crashes on short histories (length 1) or missing type_text
      Build 3 sample histories of different lengths and confirm all pass.

  (b) Honest held-out evaluation. Report metrics ONLY on a test split you did
      NOT train on. Do not report training-set or in-sample numbers. With only
      a few hundred sequences an over-parameterised model WILL overfit: a high
      score on a tiny validation split is not trustworthy. Prefer simple,
      regularised models; use k-fold cross-validation; if your held-out macro-F1
      is below the majority-class baseline, your model is too complex --- simplify
      it (fewer parameters, stronger regularisation) rather than trusting the
      inflated number. Write the real held-out accuracy + macro-F1 to
      notes/approach.md.

Step 8 --- Signal completion
  Create pretrain_done.txt with a brief summary INCLUDING your honest held-out
  accuracy + macro-F1. Only create this file AFTER step 7.5 passes.
  This file signals the evaluation harness that training is complete.

------------------------------------------------------------
OUTPUT REQUIREMENTS (all required before pretrain_done.txt)
------------------------------------------------------------

  predict.py          --- inference script (interface above)
  models/             --- saved model weights / parameters
  notes/approach.md   --- documentation of the approach
  pretrain_done.txt   --- completion signal

------------------------------------------------------------
QUALITY BAR
------------------------------------------------------------

This is a research-quality modelling task, not a quick demo.
Be thorough. Prototype, validate, iterate.
The model will be evaluated against published baselines on held-out test sequences.
Your goal is to beat those baselines.

Begin with Step 1 (EDA). Start by reading the training data.
```

Domain	Time unit	#Types	Label shown?	Event-type vocabulary (anonymised to `type_K`)
Wikipedia	hours	6	yes	content_addition, content_removal, copy_edit, minor_edit, reference, revert
GDELT	days	7	yes	armed_conflict, coercion, consultation, diplomatic_engagement, disapproval, material_cooperation, public_statement
Earthquake	days	8	yes	8 broad geographic regions (Atlantic-Africa, NW-Pacific, North-America, ...)
NBA	minutes	9	**no**	foul, free_throw, made_2pt, made_3pt, missed_fg, rebound, steal_or_block, timeout, turnover
GitHub Repo	hours	8	yes	comment, issue_closed, issue_opened, pr_merged, pr_opened, pr_reviewed, push, release
GitHub User	hours	8	yes	(same vocabulary as GitHub Repo)
Amazon	weeks	10/cat.	yes	*per-category* sub-category (10 each for Beauty, Books, Electronics, Home&Kitchen)

Table 6: The seven DESCRIBEEVENT domains. "Label shown?" is `prepend_semantic_label`: whether the real type name is prepended to each event's description in the S1/S3 history (NBA hides it, forcing symbolic-only reasoning). Amazon uses a per-*category* vocabulary: each sequence carries a top-level category (Beauty & Personal Care, Books, Electronics, or Home & Kitchen) that selects its 10 valid sub-category types.

H.3 S3 — HYBRID (REASON WITH THE S2 MODEL AS A TOOL)

S3 is *exactly the S1 prompt* with one extra block inserted inside the `<system>` section — i.e. the assembled string is `<system>{S1 system block}+{tools block}</system>` followed by the same user block — so the S1-vs-S3 comparison isolates the effect of inference-time tool access alone. The tools block exposes the S2 `predict.py`, its build notes, and the training data, plus a `<<TYPE_MAPPING>>` table (S3 reasons in the anonymised `type_K` space, but `predict.py` emits original names). The **active** variant below is the standard one used in the paper; a **base** variant (used only for the deferral ablation) offers the same resources but merely as an optional convenience ("you may call it, ignore it, or override it"), which led the model to copy the tool ∼96% of the time — the active variant's predict-first, anti-anchoring scaffold is what fixes that. `<<HISTORY_JSON_PATH>>` is the path where the current history is saved for the tool.

Active tools block (standard).

```
--- ADDITIONAL CAPABILITIES -- a FLOOR for your process, not the whole task ---
You have full bash access in a workspace with a pre-trained statistical model for this exact task, its
    ↪ build notes, and the training data. Your goal is the single BEST prediction for THIS sequence. The
    ↪ task is NOT "run the model and copy it" -- it is to reason your way to the best answer. At MINIMUM
    ↪ you should:

  1. FIRST form your OWN prediction by reasoning from the history, the event descriptions, and your world
     ↪ knowledge -- commit to it BEFORE looking at the model, so the model cannot anchor you.
  2. THEN query the pre-trained model as ONE additional piece of evidence:
        python predict.py --history_json <<HISTORY_JSON_PATH>>
        -> prints {"event_type": "<NAME>", "<<TIME_KEY>>": <number>} (ORIGINAL names; map to type_K with
     ↪ the table below).
  3. THEN decide for THIS sequence, weighing both honestly. The model captures statistical regularities
     ↪ from the training data; your own reasoning brings world knowledge and the specific context of this
     ↪ sequence. Either can be the better guide on a given case, and neither is guaranteed correct -- so do
     ↪ not reflexively adopt the model's answer OR your own first guess. Judge which to trust here, and why.

These three steps are the MINIMUM. Do MORE whenever it sharpens THIS prediction:
  - read ./notes/approach.md (the model's method + known limitations);
  - query ./training_data on the fly (base rates, transition counts, analogues);
  - run scratch-pad calculations, or build a quick auxiliary model, to test a hypothesis about this
     ↪ specific sequence.

Mapping (predict.py original name = type_K label):
<<TYPE_MAPPING>>

When done, give your final answer in the response format specified above:
brief reasoning, then EXACTLY ONE JSON line using a type_K event_type.
```

H.4 DATASET DESCRIPTIONS (DROP-IN)

Table 6 summarises the seven DESCRIBEEVENT domains. For each, the *domain blurb* is the deliberately non-leaky identity handed to the S2 builder (`<<DOMAIN_DESCRIPTION>>` above) — it states only what the data *is* (visible in the data), never any heuristic or prior, which S2 must derive itself. The same event vocabulary is anonymised to `type_K` in the S1/S3 history serialization.

Domain blurbs (verbatim).

Wikipedia Wikipedia article editing. Each sequence is a timeline of edits to a single article; types encode the nature of the edit (content additions/removals, minor edits, copy editing, references, reverts). Time is in hours between edits.

GDELT GDELT geopolitical events. Each sequence is a timeline of interactions between two nations/actors; types encode the nature of the interaction (armed conflict, diplomacy, cooperation, public statements, coercion, etc.). Time in days.

Earthquake Earthquake events. Each sequence is a timeline of earthquakes; the type is the broad geographic region of the quake, and descriptions carry per-event details (e.g. magnitude / location text). Time in days.

NBA NBA play-by-play. Each sequence is a timeline of in-game events; types encode play outcomes (made/missed shots, rebounds, fouls, free throws, turnovers, steals/blocks, timeouts). Time in minutes. (Labels hidden — symbolic only.)

GitHub Repo GitHub repository activity. Each sequence is a timeline of developer actions on a single repository (pushes, comments, pull-request and issue activity, releases). Time in hours.

GitHub User GitHub user activity. Each sequence is a timeline of a single developer's actions *across* repositories (same action vocabulary). Time in hours.

Amazon Amazon product reviews. Each sequence is a timeline of reviews written by a single user within one top-level category; the type is the product sub-category reviewed, and descriptions carry the item, star rating, and review text. Time in weeks.

I S2 Model Builds

Recall that S2 runs the LLM *once* per domain as an agentic session that must compile whatever it knows into a shipped `predict.py` (no LLM at inference, §H.2). Because the session keeps durable notes and leaves behind the actual model code, we can report both *what* it built — the model class, its feature representation, and how many features encode what — and *why*, in the agent's own words. Each domain below gives the **model**, its **feature representation** (with counts, grouped by the signal each group captures), the **world-knowledge features** that turn a piece of domain understanding into a computable input, and the **time** model. A recurring finding about the description channel closes the section (§I.8).

I.1 Wikipedia

Model. Type $= \arg\max\big(0.4 \cdot \text{HistGBClassifier} + 0.6 \cdot \text{LSTM}\big)$ over the six edit types: a gradient-boosted tree (400 trees, lr 0.05, 31 leaves, `class_weight=balanced`) blended with a 1-layer LSTM (hidden 64, dropout 0.2). Time is a quantile gradient-boosted regressor (q=0.55) on $\log(1+\Delta t)$.

Features. The GBM reads a **57-dimensional** aggregate of the history: 18 recency type one-hots (last / 2nd-last / 3rd-last event $\times$ 6 types), 7 composition features (per-type fractions + log history length), 5 time/pace features (last, previous, mean, last-5 mean and std of the log gaps), 2 recent "edit-war" rates (fraction of `revert` and `content_removal` in the last five events), a **14-feature parsed-text block**, 7 recent-window (≤ 5) averages of those text flags, a magnitude$\times$removal interaction, and 3 editor-continuity features (same editor as the last two edits, current editor's consecutive-run length, number of distinct editors in the last five). The LSTM consumes a 22-d per-event vector (6 type one-hots + log gap + the 14 text flags + a same-editor flag). The 14 text flags regex-parse the templated edit summary into edit-war and content signals: anonymous-editor, vandalism, and revert-mention flags; section and references-section flags; add-source/citation; and edit magnitude (small$\rightarrow$large), is-removal, signed log byte-delta, and percent-of-article changed.

World-knowledge features. These features are exactly where the LLM's model of Wikipedia enters: "edit wars cluster" $\rightarrow$ vandalism/revert flags + recent revert rate; "big removals get restored" (`content_removal`$\rightarrow$`content_addition` $= 0.48$, the strongest off-diagonal) $\rightarrow$ magnitude/is-removal/byte-delta + the interaction term; "reference bursts" $\rightarrow$ references-section and add-source flags; "editor continuity" $\rightarrow$ the three editor features. Class-balancing the GBM is itself a prior — it trades accuracy for minority (`revert`) recall to match the macro-F1 objective. The text block adds $+0.027$ macro-F1 over a 32-d no-text ablation.

Time. Quantile GBM at q=0.55 ($\approx$ conditional median, MAE-optimal for the heavy right tail) on $\log(1+\Delta t)$, inverted with $\exp - 1$.

I.2 GDELT

Model. Type $= \arg\max\big(\text{LogReg}(C{=}0.5)\ \text{posterior}/\text{prior}^{0.7}\big)$ over the seven interaction types; time is a gradient-boosted regressor with absolute-error loss (300 trees, lr 0.05, depth 3).

Features. The classifier reads a **42-dimensional** vector: 28 multi-window type histograms (proportion of each of the 7 types over the last $3/5/10/20$ events), 7 full-history type proportions, the 3 previous type ids (short-range order), and 4 pace features (log of the last gap, log mean of the last 3 and last 10 gaps, position index). The time head reads a separate 9-d vector of recent-gap medians/means and a recent-minimum gap. Notably, the descriptions are *not parsed at all* — `featurelib.py` never touches `type_text`.

World-knowledge features. GDELT is the honest contrast to Wikipedia: the CAMEO taxonomy, quad classes and Goldstein intensity were considered but the text turned out to be a templated restatement of the label, so the distilled prior here is *structural, not lexical*. Two structural choices carry it: the **prior-correction exponent** ($\text{posterior}/\text{prior}^{0.7}$) encodes "under a near-marginal next-type signal, macro-F1 is maximised by covering minority classes" (lifting macro-F1 from 0.06 majority to 0.181), and the multi-window histograms plus pace features encode the episodic burstiness of conflict activity (log-gap autocorrelation 0.35).

Time. Absolute-error GBM ($\approx$ conditional median) on recent-gap features, predicting raw days; MAE 7.88 vs 8.65 for the best constant.

I.3 EARTHQUAKE

Model. Zone $= \arg\max\big(\text{RandomForest}(300, \text{depth } 14)\ \text{prob}/\text{prior}^{0.6}\big)$ over the eight regions; time is an `ExtraTreesRegressor` (250, depth 14) on $\log \Delta t$, geometrically shrunk toward the global median ($0.6 \cdot \hat{y} + 0.4 \cdot \log \text{median}$).

Features. A single **98-dimensional** vector: 32 running zone-distribution features (full-history + last-$3/5/10$ fractions over 8 zones); the 27 **magnitude-weighted, time-decayed self-excitation ("Hawkes") scores** that are the model's top-ranked features; 16 last-event zone one-hot + per-zone recency; 7 parsed description fields (magnitude, depth, severity, significance, azimuthal gap, station-count present, log stations); 7 recent-magnitude and aftershock-potential stats (recent max/mean magnitudes; events and days since the last $M \geq 6$); 5 inter-arrival statistics; and 4 sequence-context features (log length, log elapsed, event rate, zone entropy). The description is regex-parsed for magnitude (`M5.3`), depth, a severity word (Minor→Great), station count, azimuthal gap, and significance score.

World-knowledge features. The 27 Hawkes scores *are* the seismology prior: for each zone z and timescale $\tau \in \{0.05, 0.3, 2.0\}$ days, $\text{score}_z(\tau) = \sum_{i \in z} \exp(\min(\text{mag}_i - 5, 3)) \cdot \exp(-(t_{\text{now}} - t_i)/\tau)$. The $\exp(\text{mag}-5)$ factor is the **Utsu/ETAS productivity law** (a larger mainshock triggers exponentially more aftershocks), and $\exp(-\text{age}/\tau)$ is the **Omori-law** temporal decay across three timescales (rapid bursts to slow regional activity). Together they map "a recent large event in a zone makes that zone the likely next zone" into a per-zone intensity; the "days since the last $M \geq 6$" features add aftershock-productivity persistence. The description-derived features add $+0.021$ macro-F1.

Time. `ExtraTrees` on $\log \Delta t$, then shrunk $0.6/0.4$ toward the log global median ($\approx$ 0.27 d) — the robust MAE minimiser given a weak conditional time signal.

I.4 NBA

Model. A 2-layer GRU (hidden 128, dropout 0.2) over per-event feature vectors, with a 9-way type head and a time head that reads the GRU state concatenated with the *detached* type logits; trained with class-weighted cross-entropy $+ 5\times L_1$ on Δt.

Features. A **40-dimensional per-event vector**: 9 type one-hots, 26 structured play flags parsed from the text, 1 shot distance, 2 possession flags (same-team-as-previous / possession-change), and 2 time features (Δt and $\log(1{+}\Delta t)$). The 26 play flags are the distilled rulebook: made/missed; a free-throw-set counter (number, total, is-last, made); rebound offensive/defensive; 6 foul-subtype flags (shooting / personal / offensive / loose-ball / technical / other); 7 turnover-subtype flags; steal and block; shot type (2pt/3pt); and an "and-1" flag.

World-knowledge features. Each flag encodes a rule a plain type+time model cannot see: made/missed encodes the $\sim$90% missed-FG→rebound rule; the six foul flags encode that a *shooting* foul sends a player to the line while other fouls do not (flipping next-event $P(\text{free_throw})$ from 94% to 17%); the "X of Y" free-throw counter (present *only* in the text) encodes that a non-final free throw is followed by another; rebound off/def encodes the possession flip; and turnover/steal flags encode the steal logged at $\Delta t{=}0$. Because time is type-determined (`free_throw` and

steal_or_block always log Δt=0), the time head conditions on the detached type logits — wiring "predict the type and you have largely determined the time" into the architecture. These features add $+0.06$ macro-F1 and cut time MAE 14% over an identical model without them.

Time. Raw minutes (no transform), L_1-trained ($5\times$ weight) $\approx$ conditional median, conditioned on the predicted type; Δt=0 is a valid target (24% of gaps are exactly zero).

I.5 GitHub Repo

Model. A **5**-member ensemble of a 2-layer LSTM (hidden 128, dropout 0.2) that consumes the whole history as a sequence and predicts at the last position; type = argmax of the mean member softmax over 8 types, time = median of the members' log-time outputs. Trained with inverse-frequency$^{0.4}$ class-weighted cross-entropy $+ 0.5\times$ L_1 time loss.

Features. A **27-dimensional per-event vector** = 8 type one-hots + $\log(1{+}\Delta t)$ + 18 **distilled text features**: 3 comment-channel flags (inline code-review / PR-thread / issue-discussion), 2 review-polarity flags (approved vs. changes-requested), 3 actor flags (is-bot, dependabot, github-actions), 8 lifecycle/automation keyword cues (stale, close, merge, scheduled "daily" doc PR, dependency-bump, CI/doc/SDK tag, a #NNN PR/issue reference, co-authored-by), and 2 length/emptiness features.

World-knowledge features. The bare type sequence (fed to the LSTM) already carries the PR lifecycle (pr_opened→review→pr_merged→ push, with pr_merged→push = 0.72); the text features add what the label hides — review *semantics* (approved ⇒ merge-imminent vs. changes-requested ⇒ more commits), comment *intent* (code review vs. PR discussion vs. issue triage), and *bot-burst determinism* (the is-bot/dependabot/ github-actions flags plus the stale/close/merge/daily cues detect the templated automated workflows that sharply narrow the next event).

Time. L_1-trained conditional median of $\log(1{+}\Delta t)$; the ensemble takes the member median, then $\exp -1$ (MAE reported as near-irreducible).

I.6 GitHub User

Model. Two gradient-boosted-tree models over one *pooled* feature vector (not a sequence): a HistGradientBoostingClassifier with sqrt-balanced class weights for type and a quantile (q=0.5) HistGradientBoostingRegressor on $\log(1{+}t)$ for time.

Features. A **72-dimensional** pooled summary: the 3 previous type ids; 24 type-count fractions (8 types $\times$ windows $\{5, 10, \text{all}\}$); 7 timing features (log last / median-of-3 / mean-of-5 / history-median gaps, a burst fraction of sub-minute gaps in the last 5, an events-within-1h count, log length); 4 repo-continuity features (same-repo-as-last over the last 3, distinct repos in the last 5, returning-to-a-known-repo, repo-parseable); and 17 semantic flags of the last event plus the same 17 averaged over the last three. The 17 flags are review polarity (approved, changes-requested), comment subtype (review-comment, on-issue, on-PR, has-PR-number), and 10 conventional-commit keyword flags (fix, feat, chore, deps, docs, refactor, test, ci, fixup/WIP, style), plus a length. Crucially the repo *identity* is never a feature (it would not generalise) — only the relational continuity features are.

World-knowledge features. PR lifecycle (recent type ordinals + the 24 type fractions), review decision (approved vs. changes-requested), *repo stickiness* (the four continuity features turn the parsed repo into "are we still in the same task?"), bursty cadence (the sub-minute burst fraction and 1h count), and kind-of-work cues (the conventional-commit keywords, with deps/bump marking automated dependency PRs). The text/repo features add $+0.009$ macro-F1 over a type+time model — a modest, honestly-reported lift.

Time. Quantile-0.5 (conditional median) GBM on $\log(1{+}t)$.

I.7 Amazon

Model. No neural network: a small JSON artifact plus a ∼50-line scorer. The type score for each candidate sub-category t (restricted to the inferred category's 10 valid types) is

$$\text{score}(t) = w_r \cdot \frac{\text{rw}(t)}{\sum_{t'} \text{rw}(t')} + w_m \cdot \frac{\text{trans}[c\|\text{last}][t] + \alpha\, \text{prior}[c][t]}{\text{trans_sum}[c\|\text{last}] + \alpha},$$

with w_r=1, w_m=3, recency decay 0.85, add-α=0.5 Markov smoothing backing off to the category prior; the top-level category is inferred by majority vote of the observed sub-categories.

Features / signals. Three signals only: (i) a per-user recency-weighted sub-category frequency (decay 0.85, recomputed each call), (ii) a smoothed first-order Markov transition $P(\text{next} \mid \text{last})$ held per category, and (iii) the category prior used both as the Markov backoff and as the standalone predictor when the context is unseen or the history empty. The shipped artifact stores only the per-category priors (raw counts) and 40 Markov rows (`"category‖last"`); everything user-specific is recomputed from the input. The review *text is deliberately not used* — "the only reliably predictive content in `type_text` is the sub-category name, which is already the event label, so it is redundant," and a TF-IDF text model "added $\sim$0 in an ensemble."

World-knowledge features. "Personal preference dominates" under $\sim$50-event histories $\rightarrow$ the recency-weighted frequency term; "same-session topical bursts / persistence" $\rightarrow$ the recency weighting (max weight on the last event) $+$ the Markov term; "sub-category co-occurrence" $\rightarrow$ the per-category transition matrix. The choice of a deliberately simple, low-variance model is itself a reasoned finding: "learned / over-parameterised models overfit — with only $\sim$150 training sequences per fold they fail to beat simple personalised models and collapse toward the majority class" (a logistic model scored 0.113 and a boosted-tree 0.227, both below the 0.246 recency baseline).

Time. The per-user median inter-arrival gap (median-of-gaps, MAE-optimal for the strongly bimodal distribution — 58% same-session bursts plus a heavy tail); fallback 0.5 weeks.

I.8 A RECURRING FINDING: THE DESCRIPTION CHANNEL IS LARGELY COMPRESSIBLE

Across all seven domains the S2 agent independently reached the same conclusion about the natural-language descriptions that nominally distinguish DESCRIBEEVENT from a plain TPP: the templated text is *largely redundant* with the event-type label, so the world knowledge worth keeping can be *distilled into a handful of regex/structured features* (or, for GDELT and Amazon, dropped entirely) rather than carried by a Sentence-BERT encoder or any inference-time LLM. The size of the residual text signal varies — meaningful on NBA ($+0.06$ macro-F1, via foul subtype and free-throw counters) and on Wikipedia ($+0.027$) / Earthquake ($+0.021$) / GitHub, but near-zero on GDELT and Amazon. This is exactly the "can the LLM's understanding of the descriptions be compiled into a static predictor" question the S2 regime is designed to probe.

J EXAMPLE CHAINS-OF-THOUGHT

We reproduce a handful of verbatim reasoning traces, a mix of correct and incorrect predictions. Each is the model's exact free-text output ending in the JSON answer, in which event types appear in the anonymised `type_K` space (alphabetical within each domain's vocabulary, Table 6; a short legend precedes each domain). To let the reader judge whether a prediction was correct *and* whether it was reasonable, we surround each trace with the de-anonymised context: the *recent actual events* leading up to the prediction and the *true next event* with its real description. The S1 traces are the pure verbal prior (no tools); the S3 traces (§J.5) show the model reasoning *with* the S2 `predict.py` available as a callable tool.

J.1 S1 — NBA

Legend: `type_0`=foul, `type_4`=missed_fg, `type_5`=rebound, `type_6`=steal_or_block, `type_8`=turnover.

Correct — pred `missed_fg`, true `missed_fg` ($\hat{t}$=0.20, true 0.33 min). *Recent events (oldest→newest):* `foul` (technical), `free_throw` (Brunson), `made_3pt` (Brunson, 27 ft), `made_3pt` (Nurkic, 29 ft).

```
A made 3-pointer by Phoenix means the Knicks get possession next, so the next event
is likely a shot attempt. Given the recent rhythm, a missed shot (type_4) followed by
 a rebound is the most common pattern, with gaps typically around 0.15-0.25m.

{"event_type": "type_4", "time_minutes": 0.2}
```

True next: `missed_fg` — "C. Payne misses a 3-point jump from 27 ft" ($+0.33$ min). Correct type; the model underestimated the gap.

Incorrect — pred `missed_fg`, true `foul` ($\hat{t}$=0.30, true 0.41 min). *Recent events (oldest→newest):* `made_3pt` (Brooks), `missed_fg` (Hachimura), `rebound` (Thompson), `made_3pt` (VanVleet).

```
Houston just made a 3-pointer (made basket), so possession flips to the Lakers, who
have been missing many threes recently. The most common next event is a shot attempt,
 likely a miss given the recent pattern. Time gap for a full possession typically
runs ~0.25-0.35m.
{"event_type": "type_4", "time_minutes": 0.3}
```

True next: `foul` — "J. Tate commits a personal foul" (+0.41 min). A plausible "shot attempt" guess, but a foul intervened.

J.2 S1 — GitHub Repo

Legend: `type_2`=issue_opened, `type_3`=pr_merged, `type_4`=pr_opened, `type_5`=pr_reviewed, `type_6`=push.

Correct — pred `pr_merged`, true `pr_merged` ($\hat{t}$=0.015, true 0.084 h). *Recent events (oldest→newest):* `pr_merged` (egeakman, Pr6), `push`, `pr_merged` (egeakman, Pr5), `push`.

```
The recent pattern shows a tight cascade of sequential PR merges (type_3) each
immediately followed by a push (type_6), unwinding the dependent PR chain (Pr14 down
to Pr5). The last event was a push after merging Pr5, so the next is most likely the
merge of Pr4 with a very short gap.

{"event_type": "type_3", "time_hours": 0.015}
```

True next: `pr_merged` — "Pull request merged by egeakman: Pr4 update config" (+0.084 h). Correct, and the cascade it inferred (Pr5→Pr4) is exactly right.

Incorrect — pred `push`, true `issue_opened` ($\hat{t}$=2.5, true 11.5 h). *Recent events (oldest→newest):* `release` (github-actions bot, v0.0.29), `issue_opened` (sirgwain), `issue_opened` (sirgwain), `issue_opened` (tmcmh).

```
Recent events show a cluster of newly opened issues (type_2) by sirgwain and
contributors. The established pattern shows sirgwain typically batches fixes into a
push (type_6) after issues accumulate, often within a day. Given the most recent
issue was opened by tmcmh, the next event is likely continued issue activity or a fix
 push.
{"event_type": "type_6", "time_hours": 2.5}
```

True next: `issue_opened` — "Issue opened by sirgwain" (+11.5 h). The issue-filing burst simply continued; the trace even named this option ("continued issue activity") before choosing `push`.

J.3 S1 — Wikipedia

Legend: `type_0`=revert, `type_2`=copy_edit, `type_3`=minor_edit.

Correct — pred `copy_edit`, true `copy_edit` ($\hat{t}$=4.5, true 12.6 h). *Recent events (oldest→newest):* `copy_edit` (anon), `minor_edit` (Anaxial, revert), `content_removal` (anon, large), `content_addition` (Enviroboy, restores it).

```
The last event was a revert (type_0) restoring the article after vandalism. The
recurring pattern shows that after restoration, the next event is typically a small
anonymous copy_edit (type_2) after a few hours, or another vandalism cycle. Given the
 cadence, a small edit after a modest gap is most likely.

{"event_type": "type_2", "time_hours": 4.5}
```

True next: `copy_edit` — "anonymous small addition (+6 bytes)" (+12.6 h). Correct type (the removal was just restored, an anon tidy-up follows).

Incorrect — pred `copy_edit`, true `minor_edit` ($\hat{t}$=18.0, true 7.5 h). *Recent events (oldest→newest):* `copy_edit` (UndercoverClassicist), `copy_edit` (JennyOz), `minor_edit` (UndercoverClassicist), `copy_edit` (KuyaMoHirowo).

```
Page protection was just added, suggesting the article is now in a stable/finalized
state, so the next edit is likely sparse. The dominant event type throughout is
type_2 (copy_edit), often following longer gaps after pauses. Given the recent
slowdown and protection event, I expect a moderate-to-long delay before another minor
 copy edit.

{"event_type": "type_2", "time_hours": 18.0}
```

True next: `minor_edit` — "MisfitToys makes a small addition to the History section" (+7.5 h). A near-miss — `copy_edit` vs. `minor_edit` are adjacent small-edit types, and `copy_edit` was indeed dominant.

J.4 S1 — GDELT

Legend: `type_0`=armed_conflict, `type_1`=coercion, `type_2`=consultation, `type_3`=diplomatic_engagement, `type_4`=disapproval.

Correct — pred `consultation`, true `consultation` ($\hat{t}$=4, true 3 d). *Recent events (oldest→newest, Armenia–Azerbaijan):* `disapproval` (Azerbaijan rejects Armenia), `public_statement` (Armenia appeals), `armed_conflict` (Armenia uses force), `disapproval` (Azerbaijan rejects).

```
Recent events show escalating tension (rejections, armed conflict, disapprovals), but
 the dominant pattern throughout is a return to type_2 consultation/type_3 diplomatic
 engagement after friction. After back-to-back disapprovals, a consultation typically
 follows, with gaps trending in the 4-7 day range lately.

{"event_type": "type_2", "time_days": 4}
```

True next: `consultation` — "Armenia engaged in formal negotiations with Azerbaijan" (+3 d). Correct: friction did revert to consultation.

Incorrect — pred `consultation`, true `disapproval` ($\hat{t}$=15, true 28 d). *Recent events (oldest→newest, UK–Iran):* `disapproval` (Iran criticises UK), `consultation` (UK–Iran talks), `coercion` (UK sanctions Iran), `coercion` (UK sanctions Iran).

```
Recent events show escalating coercion (sanctions/embargoes), with the last two being
 rapid type_1 events. After such escalation, a consultation or diplomatic response (
type_2/type_3) often follows, though continued coercion is also common. Given the
back-and-forth pattern, a consultation is most likely with a moderate gap.

{"event_type": "type_2", "time_days": 15}
```

True next: `disapproval` — "Iran accused Britain" (+28 d). The escalation continued (verbal conflict) rather than de-escalating to talks — a reasonable but wrong bet.

J.5 S3 — REASONING WITH THE S2 MODEL AS A TOOL

These traces (GitHub Repo) show the active-calibration S3 protocol: the model forms its own view, queries `predict.py`, then decides — here mostly *agreeing* with the tool but re-calibrating the time. *Legend:* `type_3`=pr_merged, `type_4`=pr_opened, `type_6`=push.

Correct — pred `pr_merged`, true `pr_merged` ($\hat{t}$=0.01, true 0.008 h). *Recent events (oldest→newest):* `pr_opened` (bmc08gt), `pr_merged` (bmc08gt), `push` (bmc08gt), `pr_opened` (bmc08gt, localization).

```
The model agrees: 'pr_merged'. My own reasoning (bmc08gt's rapid open->merge pattern)
 and the model both point to type_3. The model's 0.048h is reasonable; given the
recent quick merges (~0.004-0.006h), I'll settle slightly below the model's estimate.

bmc08gt consistently merges his own PRs almost immediately after opening (events
50->51, 59->60 within ~0.004h), and the model independently predicts pr_merged. Both
agree on type_3; I'll use a short gap consistent with his recent merge cadence,
nudged toward the model.

{"event_type": "type_3", "time_hours": 0.01}
```

True next: `pr_merged` — "Pull request merged by bmc08gt (update localization)" ($+0.008$ h). Both type and the short gap are essentially exact.

Incorrect — pred `pr_opened`, true `push` ($\hat{t}$=0.25, true 18.5 h). *Recent events (oldest→newest):* `comment` (profvjreddi, on issue), `pr_opened` (kai4avaya, darkmode), `pr_merged` (profvjreddi, darkmode), `push` (profvjreddi).

```
Both my reasoning and the pretrained model agree on 'pr_opened'. The model predicts
0.24h. Given the kai-widget cycle just closed and this repo frequently spawns a new
PR shortly after a merge-push, I'll align with the model's type and its median-
optimal short time estimate, nudging slightly.

The model and my own reasoning converge on 'pr_opened'. The time gaps to a new PR
here are highly variable (0.1h to 23h), so I trust the model's MAE-optimal median
estimate over my higher guess.

{"event_type": "type_4", "time_hours": 0.25}
```

True next: `push` — "Code pushed by profvjreddi" (more doc changes) ($+18.5$ h). The just-closed cycle was followed by another push, not a new PR, after a long wait.

Note. The second S3 trace illustrates a characteristic failure mode of the computation-dominated time channel: both the model and the model's own reasoning commit to a short, median-optimal gap ($\sim$0.25 h), which minimises expected MAE but misses this instance's long (18.5 h) wait — the tool and the reasoner agree, and are jointly wrong.

K Testing S2 on a Specialized TPP Task - POI

The main results compare S1/S2/S3 against controlled baselines on DescribeEvent. This appendix asks whether the S2 recipe can build a predictor that holds up against task-specialised architectures on a contested benchmark. We test it on human-mobility next-venue (next-POI) prediction, an instance of TPP forecasting (here the "event type" is the venue), which has a mature leaderboard.

Protocol. We use the preprocessed New York City Foursquare split of Yang et al. (2022) (1,047 users, 4,980 venues) and evaluate on the last step of each test trajectory over in-vocabulary targets (n=1,309), the setting the leaderboard reports. The metric is Acc@1: whether the model ranks the true next venue first. To fix the protocol we reproduce GETNext and recover its reported accuracy (Acc@1 $= 0.251$ against the 0.244 of Yang et al., 2022); GETNext and S2 are then scored by that same harness, and the remaining numbers are as reported by their authors.

What S2 builds. From the training check-ins and a short domain description, the S2 agent writes a self-contained predictor that scores venues with a rank-average blend of a gradient-boosted LambdaMART ranker (a five-seed LightGBM ensemble) and a small neural listwise ranker (a GRU over the check-in history with venue, user, category, and temporal embeddings), over engineered mobility features (personal frequency and transition statistics, spatial distance-decay, temporal context, category transitions). The LLM is used only to build this model; none is called at inference.

method	Acc@1	uses at inference
GETNext (Yang et al., 2022)	0.251	—
STHGCN (Yan et al., 2023)	0.273	—
S2 (ours)	**0.280**	—
ROTAN (Feng et al., 2024)	0.311	target arrival time
LLM4POI (Li et al., 2024b)	0.337	target arrival time, an LLM
GNPR-SID (Wang et al., 2025a)	$\sim$0.36	target arrival time, an LLM
CoAST (Zhai et al., 2025)	$\sim$0.40	target arrival time, an LLM

Table 7: Next-venue Acc@1 on the GETNext NYC split (last step, in-vocabulary targets, n=1,309). GETNext and S2 share one scoring harness — our GETNext run reproduces its reported Acc@1 (0.251 vs. 0.244) — and the other numbers are as reported by their authors. The top group calls neither an LLM nor the target's arrival time at inference; each method in the bottom group adds one of those.

Takeaway. S2 builds a custom model which runs LLM-free at inference time. It improves on the tuned deep model GETNext and is on par with the specialised hypergraph model STHGCN. The methods that score higher each rely on a signal S2 does not use at inference: the target check-in's arrival time (ROTAN), a look-ahead when the task is to predict *where* a user goes next, or an LLM invoked per prediction (LLM4POI, GNPR-SID, CoAST). We read this as external evidence that the S2 mechanism produces deployable models competitive with task specialized research grade models tested on a competitive benchmark.