A Multi-Layer Cooperative Multi-Agent Framework for Time Series Forecasting with Large Language Models

Chung-Nan Tsai[1], Jingnan Xie[2], Chun-Ming Lai[3] and Ching-Sheng Lin[4]

1 Lam Research Japan GK, Kanagawa 222-0033, Japan; chung-nan.tsai@lamresearch.com

2 Computer Science, Millersville University of PA, Millersville, PA 17551, USA; jingnan.xie@millersville.edu

3 Department of Computer Science, Tunghai University, Taichung City 407, Taiwan; cmlai@thu.edu.tw

4 Master Program of Digital Innovation, Tunghai University, Taichung City 40704, Taiwan; cslin612@thu.edu.tw

Corresponding author: Ching-Sheng Lin

Abstract

Motivated by the remarkable performance of Large Language Models (LLMs) across diverse domains, we present a novel hierarchical multi-agent collaborative framework leveraging LLMs to address time series forecasting challenges. Leveraging advanced prompting techniques including Chain-of-Thought (CoT) reasoning and Optimization by PROmpting (OPRO), our system implements a multi-layer agent-based cooperative architecture comprising Forecast, Support, and Debate. The agents in the Forecast layer generate initial predictions, those in the Support layer refine them while the agents in the Debate layer perform critical evaluation. This structured cooperation allows LLMs to iteratively improve prediction quality and adapt to complex time series data. Experimental results demonstrate that our multi-layer collaborative multi-agent framework achieves promising accuracy and robustness compared to both traditional training-based models and existing LLM-based approaches in time series forecasting tasks.

Keywords Large language models, multi-agent, time series forecasting, prompting techniques, multi-layer

1. Introduction

Time series analysis serves as a foundational method for understanding sequential patterns and trends that evolve over time [1]. Recent research in this field has largely focused on classifying time series data, forecasting future trends, and detecting anomalies. Among them, time series forecasting (TSF) utilizes historical time-ordered data to predict future values and serves as a crucial foundation for decision-making in many domains including economics, infrastructure, society, and beyond [2, 3].

Machine learning and deep learning have significantly enhanced TSF by capturing complex temporal dependencies and nonlinear patterns. These models can efficiently process large-scale time series data, reducing the need for manual feature engineering and enabling more accurate long-term predictions [4, 5]. However, challenges remain, particularly in transferring learned knowledge across different datasets and handling sparse or irregularly sampled time series data [6]. Recently, Large language models (LLMs) have become essential tools and have been widely adopted across various research domains, including natural language processing, healthcare, finance, and scientific discovery. They have enabled advancements in tasks such as text generation, data analysis, and automated reasoning [7, 8]. In light of this, LLM-based approaches have also been applied to TSF tasks. Currently, most LLMs employed for TSF depend on fine-tuning (such as Promptcast [9] and LLM4TS [10]), which comes with several limitations. First, fine-tuning is computationally expensive and time-consuming, especially for large models, making it difficult to adapt to different domains efficiently. Additionally, it requires a substantial amount of training data, which may not always be available and can reduce its practicality in real-world TSF applications. Compared to those fine-tuning-based

approaches, zero-shot approaches leverage pretrained LLMs without additional task-specific training. By eliminating the need for resource-intensive fine-tuning, they offer computational efficiency, seamless domain adaptability without retraining, and enhanced data efficiency which makes them effective even in scenarios with limited historical data [11, 12]. However, since LLMs often struggle with processing long sequences which are also a common characteristic of time series data, they may fail to effectively capture complex temporal patterns in zero-shot TSF [13]. This limitation may lead to decreased accuracy and flexibility when dealing with diverse and evolving datasets.

In addition to serving as NLP task models or general-purpose chatbots, recent research on LLMs has demonstrated their remarkable ability to adapt to different roles and personas, enabling them to generate responses tailored to specific tasks and user preferences. Building on this, the concept of persona-based prompting has been widely explored in target scenarios, where an LLM assumes a distinct identity or expertise to produce more contextually accurate responses and maximize their overall performance [14]. Furthermore, multi-agent cooperation, where multiple LLM instances adopt distinct personas and interact to solve complex tasks, has emerged as a promising paradigm in enhancing decision-making and problem-solving. These approaches have been utilized in forms such as multi-perspective reasoning, debate-style discussions, and collaborative problem-solving [15]. Inspired by these advancements, we propose leveraging multi-agent cooperation to enhance zero-shot TSF. Traditional zero-shot methods often struggle to capture complex temporal dependencies due to their lack of task-specific training and difficulty in handling long sequences. By incorporating multiple LLM personas with specialized forecasting roles, our approach aims to improve forecasting accuracy and adaptability. Moreover, through structured cooperation and persona-specific expertise, the proposed method can mitigate the limitations of standard zero-shot forecasting while maintaining computational efficiency. In this study, different agents are designed with distinct personas, and therefore, the terms "agent" and "persona" will be used interchangeably.

The novelty and key contributions of our proposed method can be summarized as follows:

- We introduce a persona-driven multi-layer multi-agent framework to enhance diversity in TSF. This hierarchical approach ensures that different perspectives are incorporated across multiple layers and leads to more robust predictions.
- We design a debate-driven planning mechanism to select the best outcome from the previous layer's results. This approach enhances decision-making by refining predictions through iterative evaluation and consensus-building.
- We conduct extensive experiments on the public datasets to evaluate the effectiveness of our approach. The results demonstrate improvements in forecasting accuracy, highlighting the advantages of our proposed method.

The remainder of this paper is organized as follows. Section 2 provides a review and discussion of related technologies and relevant works. In Section 3, we present our proposed model architecture. Section 4 outlines our experimental studies and their results. Finally, Section 5 concludes the paper and discusses potential future research directions.

2. Related works

In this section, we review the related work in TSF, focusing on machine learning-based approaches and the recent application of LLMs for this task. We explore how traditional machine learning methods have evolved and how LLMs are being increasingly utilized to enhance TSF accuracy and efficiency.

Conventional TSF techniques, such as ARIMA [16] and Gaussian Processes [17], have long been essential tools in analyzing temporal patterns but struggle to capture complex nonlinear relationships and long-term dependencies in real-world data. With the emergence of deep neural network architectures, the focus of TSF research has shifted. DeepAR is an autoregressive recurrent neural network model designed for probabilistic TSF by leveraging large sets of related time series data [18]. TCN (Temporal Convolutional Network) is a fully convolutional architecture that ensures sequence length preservation using zero padding and prevents future information leakage through causal convolutions [19]. In recent years, the advancement of deep learning has led to the emergence of methods such as Multilayer Perceptron (MLP)-based and Transformer-based approaches, which have gained significant attention in TSF. N-BEATS is a deep learning-based TSF model that utilizes a fully connected neural network with residual blocks to capture trend and seasonality without relying on recurrent or convolutional architectures [20]. DLinear is a TSF model that employs a straightforward

linear layer to directly map past observations to future values [21]. The Transformer model has demonstrated significant success across multiple domains and recently expanded to TSF, as its attention mechanism effectively captures dependencies within sequential data [22]. Informer is an optimized Transformer-based model for long sequence TSF, enhancing efficiency with ProbSparse Self-Attention, self-attention distilling, and a generative-style decoder for faster inference [23]. FEDformer improves long-term TSF by integrating a seasonal-trend decomposition method to capture global patterns and applying a frequency-based transformation to process time series data in a more structured manner [24]. PatchTST introduces a Transformer-based approach for multivariate TSF by segmenting time series into patches as input tokens and applying a channel-independent design, where each univariate series shares the same embedding and model weights [25]. This design preserves local semantics, reduces computational complexity, and enables longer historical dependencies to be captured.

Building on the success of large pre-trained models and LLMs in various domains, researchers have begun to apply these advancements to TSF. Existing methodologies generally follow two approaches: one involves developing specialized foundation models for time series analysis, while the other adapts LLMs to process time series data as a form of natural language. LPTM (Large Pre-trained Time-series Models) addresses the challenge of learning from heterogeneous time series data by introducing an adaptive segmentation technique that dynamically determines the best way to partition data during pre-training [26]. This allows LPTM to achieve competitive performance compared to specialized models when fine-tuned for downstream tasks. TimesFM (Time-series Foundation Model) is a foundation model for TSF that employs a patch-based approach to segment data and predict future values from past observations [27]. It features a 200-million parameter architecture pre-trained on 100 billion time points with a decoder-only structure. Regarding the direct use of pre-trained LLMs for TSF, LLM-TIME is the first approach to apply zero-shot forecasting in this manner [11]. It treats time series data as next-token prediction by converting numerical values into tokenized digit sequences, using tokenization and rescaling methods to optimize token usage for efficient forecasting. LSTPrompt is a novel approach that enhances zero-shot TSF by using Chain-of-Thought prompting to guide LLMs through both short-term and long-term forecasting tasks [12]. It decomposes the forecasting process into sub-tasks, adjusting the prompts for each to improve adaptability and reasoning over time series data.

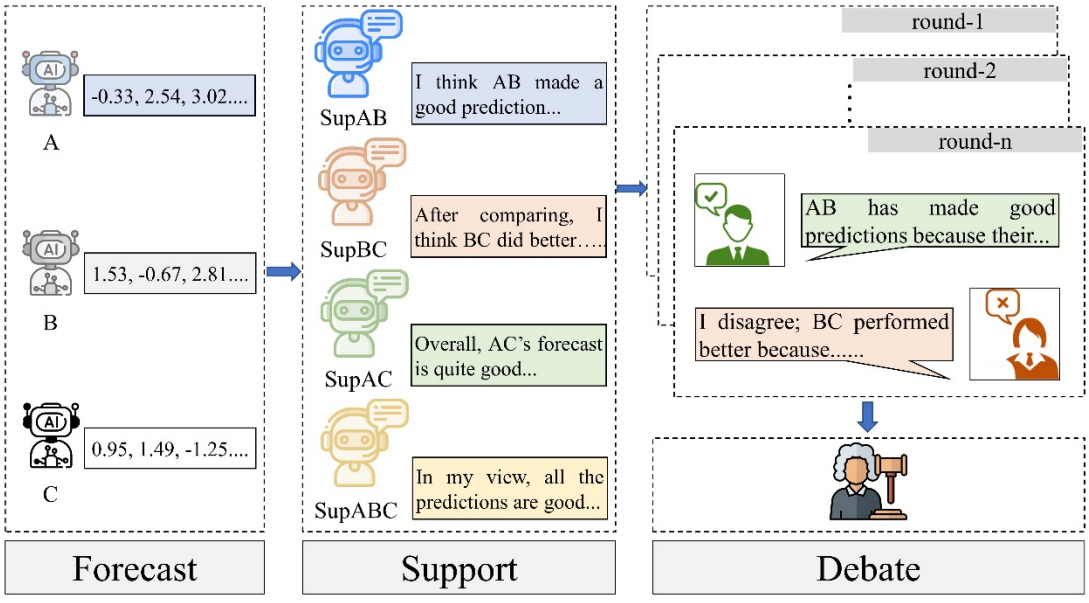


Fig. 1 The system architecture

3. Methodology

In this section, we define the TSF problem and discuss the multi-layer cooperative multi-agent LLM architecture we have designed to address it.

3.1. Problem formulation

Time series forecasting aims to predict future values with a horizon window size H based on previously observed data points with a lookback window size L. It can be formulated as follows:

$$\hat{y}_{t+1}, \ldots, \hat{y}_{t+H} = f\left(y_{t-L+1}, \ldots, y_t, Z; \theta\right) \quad (1)$$

where y represents data values, Z is the additional information which may optionally be required and θ denotes the model parameters. In this paper, we mainly investigate univariate TSF to examine how our model performs in a zero-shot setting, where the LLM parameters θ remain fixed, and Z is adjusted through techniques in prompt engineering.

3.2. Proposed framework

Our multi-layer cooperative multi-agent large language model architecture, as shown in Figure 1, consists of three key modules: Forecasting, Supporting, and Debating. The Forecasting module generates future estimates based on given time series data, the Supporting module selects majority-based subsets of previous predictions to support its reasoning, and the Debating module utilizes the supporting results from the previous stage to determine the final outcome through a debate-like process. Our approach leverages prompt-based techniques grounded in Chain-of-Thought (CoT) reasoning [28] (such as step-by-step analytical prompts), and Optimization by PROmpting (OPRO) [29] (such as " Take a deep breath "). These techniques aim to enhance the model's reasoning capabilities and task-specific alignment.

Existing research primarily relies on single-agent models, which generate forecasts based on a singular perspective. Since such an approach may be limited in capturing diverse reasoning paths and mitigating biases, we adopt a multi-agent mechanism to collaboratively perform the prediction task to address these limitations in the Forecasting module. This approach offers several advantages: (1) it enables diverse reasoning by allowing different agents to contribute unique perspectives; (2) it promotes specialization and leads to a more comprehensive prediction process; and (3) it supports a consensus-driven mechanism to enhance the robustness and reliability of the final forecast. Following this, we introduce the three persona agents: the general forecaster, who focuses on recent patterns and direct extrapolation; the pattern recognizer, who detects complex and subtle relationships in the data; and the trend analyst, who focuses on long-term trends and periodic patterns. There are two types of prompts to guide the LLM's behavior: persona-specific prompts and the general task prompt. The persona-specific prompts (shown in the top of Table 1) are tailored to each individual persona, defining their role, characteristics, and perspective to ensure that each agent provides relevant input. The general task prompt (shown in the bottom of Table 1), on the other hand, focuses on the purpose of the task and ensure that the LLM adheres to the correct methodology. We directly use the prompts from prior work [12] as our general task prompt. By combining both the persona prompts and the general task prompt, we provide the LLM with a comprehensive input that includes both the role-specific context and the task-specific instructions to ensure more accurate predictions. It is worth noting that in Figure 1, to simplify notation, we use A to represent the general forecaster, B to represent the pattern recognizer, and C to represent the trend analyst.

Table 1 The prompts for Forecasting module

General Forecaster Prompt	Pattern Recognizer Prompt	Trend Analyst Prompt

You are a Time Series Forecaster. The user will provide a sequence of decimal values separated by commas, each representing a time step. Your task is to predict the next { horizon_window_size } time steps, one step at a time. Use the provided sequence and any recent patterns to generate accurate predictions.	You are a highly detailed Time Series Forecaster with advanced pattern recognition capabilities. The user will provide a sequence and you will predict the remaining sequence. The sequence is represented by decimal values separated by commas. Each item separated by commas is a time step. You need to accurately predict the next { horizon_window_size } time steps.	You are a statistical Time Series Forecaster specializing in trend analysis and seasonal variations. The user will provide a sequence which is represented by decimal values separated by commas. You need to identify long-term trends, cyclic behavior, and anomalies to improve prediction accuracy for the next { horizon_window_size } time steps.
General Task Prompt		
Please continue the following input sequence by addressing the task of forecasting { dataname }. You should break down the task into short - term and long - term predictions, following a three - step plan. First, adaptively and reasonably identify the ranges for short - term and long - term predictions. Then, design distinct and correct forecasting mechanisms for both short - term and long - term prediction tasks. For short - term predictions, focus on trends and the last few steps of the input sequence. For long - term predictions, emphasize cyclical patterns and statistical properties of the entire input sequence. You may further optimize the forecasting mechanisms based on your observations and domain knowledge. Finally, correctly implement the forecasting mechanisms, completing predictions one - time step at a time. Remember to take a deep breath after every { breath_steps } time steps of prediction. The input sequence is as follows:		

After the individual predictions are generated in the previous stage, we introduce a Supporting module with supporting agents to enhance the reliability and robustness of the final prediction. These agents operate by selecting majority-based subsets of previous predictions for further reasoning. Specifically, given the initial forecasts produced by the three persona agents, four supporting agents are assigned to independently assess and endorse different subsets of these predictions: one supporting agent considers the predictions from the general forecaster and pattern recognizer, another from the general forecaster and trend analyst, a third from pattern recognizer and trend analyst, and the last from all three. This structured approach allows for a more comprehensive synthesis of information by incorporating multiple perspectives while maintaining diversity in decision-making. Table 2 displays the persona along with its corresponding prompt. We refer to each subset as a group ({ group_name }).

Table 2 The prompts for Supporting module

You are a decision support expert. Summarize responses from agents in group { group_name } and generate supportive insights from this group. ### Instructions for support: ### - Identify overall trends (increasing, decreasing, stable, fluctuating). - Compare agent responses: highlight agreements and major differences. - Focus on general patterns rather than listing all numbers. ### Raw agent responses for { group_name }: ### { group_name_predictions }

In the Debating module of our approach, we introduce a debate mechanism to further refine the final decision. Two debater agents engage in discussions and evaluate which of the supportive analyses is the most compelling. This process can involve multiple rounds of debate, which allow for more in-depth reasoning and comparison. We employ a two-round debate in this paper. Finally, a judge agent makes the ultimate decision and selects the most robust conclusion based on the arguments presented. This structured process enhances the reliability and transparency of the final prediction. We present the personas of the debater and the judge, along with their corresponding task prompts, in Table 3.

Table 3 The prompts for Debating module

Debater Prompt

You are an AI debater engaged in a debate with another AI debater. Your arguments should be formulated based on the supportive reports from each group in the previous stage, as well as the debate history listed below. Use these sources to construct well-reasoned and compelling arguments. ### Debate instructions: ### - Analyze the debate history. - Identify which group captures the most accurate and complete insights. - Defend your position logically and clearly. ### Reference Materials: ### Supportive reports from prior module: { supportive_analysis } The debate history: { debate_history }
Judge Prompt
You are the final judge responsible for evaluating the arguments in the debate presented below and determining which group is the most accurate and well-supported in the forecasting. ### Instructions to determine the best group: ### - Read the debate reports carefully. - Compare the reasoning, evidence, and clarity of each debater. - Identify which group is the most reliable and comprehensive. - Return ONLY the best group name without any extra text. ### Debates: ### { debate_history }

The process of our multi-layer cooperative multi-agent framework for TSF is detailed in Algorithm 1. The input consists of training data (TD) and a prompt pool (PP), which stores keys (character names) and their corresponding prompts. The output is the forecasting result. For the forecasting module, from lines 3 to 5, each forecasting agent generates predictions based on their role, task description, and training data. In the supporting module, from lines 7 to 9, each group agent generates supportive reports based on their role and the forecasting module's prediction results. Within the Debating module, from lines 11 to 18, two debaters engage in N rounds of debate, drawing from their roles, the supportive reports provided by the Supporting module, and the debate history. Finally, the judge will make a decision according to the debate history.

Algorithm 1: Multi-Layer Cooperative Multi-Agent Framework

Input: Training data: TD, prompt pool: PP
Output: Time series prediction $\hat{y}^{test}$.

N=2
#Forecasting module
for each forecast_agent:
  response(forecast_ agent) = LLM(PP(forecast_ agent), PP(general_task_prompt), TD)
end for
#Supporting module
for each group_agent:
  supportive_report(group_agent) = LLM(PP(group_agent), response)
end for
#Debating module
debate_history = []
debater1 = PP(debate_agent)
debater2 = PP(debate_agent)
for i=1 to N:
  argument1 = LLM(debater1, supportive_report, debate_history)
  argument2 = LLM(debater2, supportive_report, debate_history)
  debate_history = debate_history + argument1 + argument2
end for
#Decision
judge = PP(judge_agent)
$\hat{y}^{test}$ = LLM(judge, debate_history)
return $\hat{y}^{test}$

4. Experiments

This section presents the experimental design and analysis of our proposed approach. We begin by introducing the datasets and evaluation metrics used in the study, followed by a detailed presentation and discussion of the main experimental results. Finally, we explore extended experiments to further examine the generalizability and robustness of our method.

4.1. Dataset and evaluation metric

The Darts library provides several built-in time series datasets commonly used for benchmarking forecasting models [30]. Datasets like AirPassengers, AusBeer, Sunspots, and MonthlyMilk cover various domains. These datasets help evaluate model performance on real-world seasonal and trend-based data. We also use these four datasets in our experiments with detailed statistics in Table 4.

In this study, we employ mean absolute error (MAE) as the evaluation metric, which is widely used in the TSF literature. MAE quantifies the mean absolute differences between predicted and actual values, and is formally defined as:

$$MAE = \frac{1}{n}\sum_{t=1}^{n} |y_t - \hat{y}_t| \tag{2}$$

where $\hat{y}_t$ and y_t represent the predicted and the actual observed values, respectively, for the target series at time t.

Table 4 Overview of datasets in DARTS

Dataset	Frequency	Start	End	Length
AirPassengers	Monthly	1949/01	1960/12	144
AusBeer	Quarterly	1956/Q1	2008/Q3	211
Sunspots	Monthly	1749/01	1983/12	705
MonthlyMilk	Monthly	1962/01	1975/12	168

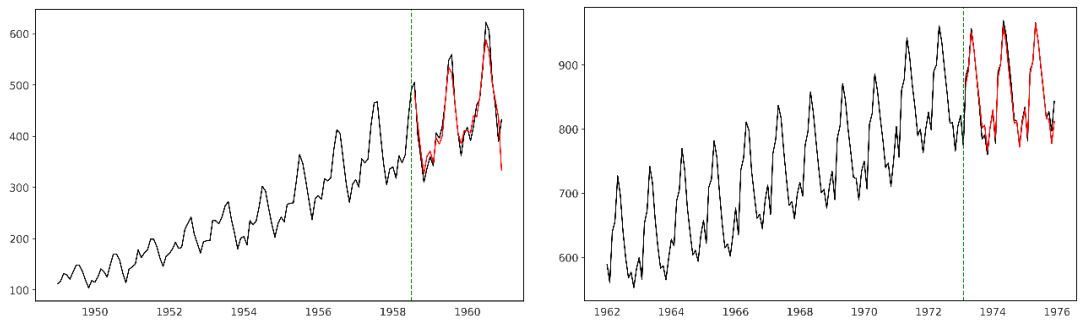
4.2. Performance evaluation and analysis

To validate the feasibility of our proposed model, we compare it with both supervised and zero-shot methods with H time steps horizon. The supervised methods include Gaussian Processes (GP), TCN, and N-BEATS, while the zero-shot methods comprise TimesFM, LLMTime, and LSTPrompt.

The results of these comparisons are presented in Table 5, where the other model results are sourced from the original paper [12]. We have the following observations. First, our model achieves the best results in three out of four comparisons. Our MAE performance has improved by 3.61% on AirPassengers, 5.71% on MonthlyMilk, and 0.43% on Sunspots. The only underperforming dataset is AusBeer, where our model shows a 2.34% higher error rate compared to the best-performing TimesFM model. Second, overall, the zero-shot methods perform better than the supervised methods. This suggests LLMs' adaptability to diverse time series patterns. Third, our model, along with other SOTA methods, exhibits relatively high MAE on the Sunspots dataset. To further analyze this phenomenon, Figure 2 presents our model's prediction results across all four datasets. From the figure, we can observe the distinct characteristics of Sunspots, including its long-term irregular cycles and fluctuating trends, which make it more challenging for models to achieve accurate predictions.

Table 5 Experimental results of different models on the Darts datasets. The top-performing results are shown in bold, while the second-best results are underlined

Dataset	H	Supervised Methods			Zero-Shot Methods			
		GP	TCN	N-BEATS	TimesFM	LLMTime	LSTPrompt	Ours
AirPassengers	29	34.67	54.96	97.89	14.75	48.96	<u>13.02</u>	**12.55**
MonthlyMilk	34	30.33	70.86	33.64	22.46	63.15	<u>7.71</u>	**7.27**
AusBeer	43	102.05	30.9	<u>10.39</u>	**10.25**	20.85	13.29	10.49
Sunspots	141	53.74	51.82	73.15	50.88	59.91	<u>46.84</u>	**46.64**



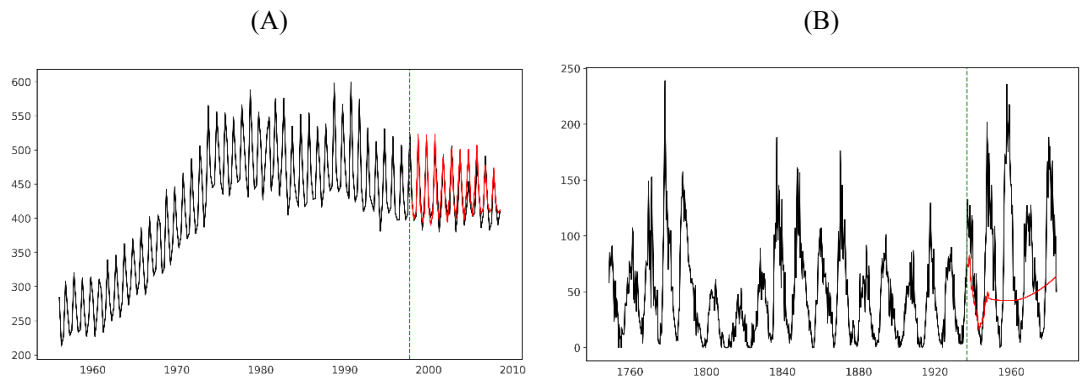


Fig. 2 Visualization of prediction results for four datasets: (A) AirPassengers, (B) MonthlyMilk, (C) AusBeer, and (D) Sunspots. The black line represents the original data, while the red line shows the predictions made by our model. The x-axis represents the time index, and the y-axis represents the corresponding values of the time series

To further analyze our model's performance under different conditions, we examine the impact of temperature on the LLM for TSF. Since LLMs inherently introduce stochasticity, temperature controls the degree of randomness in the generated outputs. Lower temperatures lead to more deterministic predictions, while higher temperatures increase variability, which may reduce consistency in forecasting tasks. We initially set the temperature to 0.200 as a baseline and subsequently test additional values of 0.150, 0.175, 0.200, 0.225, and 0.250 to evaluate their impact on model performance. Each temperature value is repeated in three experiments. The average results of different temperature, presented in Figure 3, illustrate the model's behavior under different temperature settings. We can see that the performance remains relatively consistent, indicating that temperature variations within this range have minimal impact. We further conduct an ANOVA test, and the results showed a p-value of 0.0166 for the AirPassengers data, 0.0451 for the MonthlyMilk data, 0.6446 for the AusBeer data, and 0.1545 for the Sunspots data. Based on a significance threshold of 0.01, neither the AirPassengers nor the MonthlyMilk datasets show statistically significant differences, as their p-values exceed the 0.01 threshold. Additionally, the AusBeer and Sunspots datasets also do not exhibit significant differences, with p-values well above 0.01.

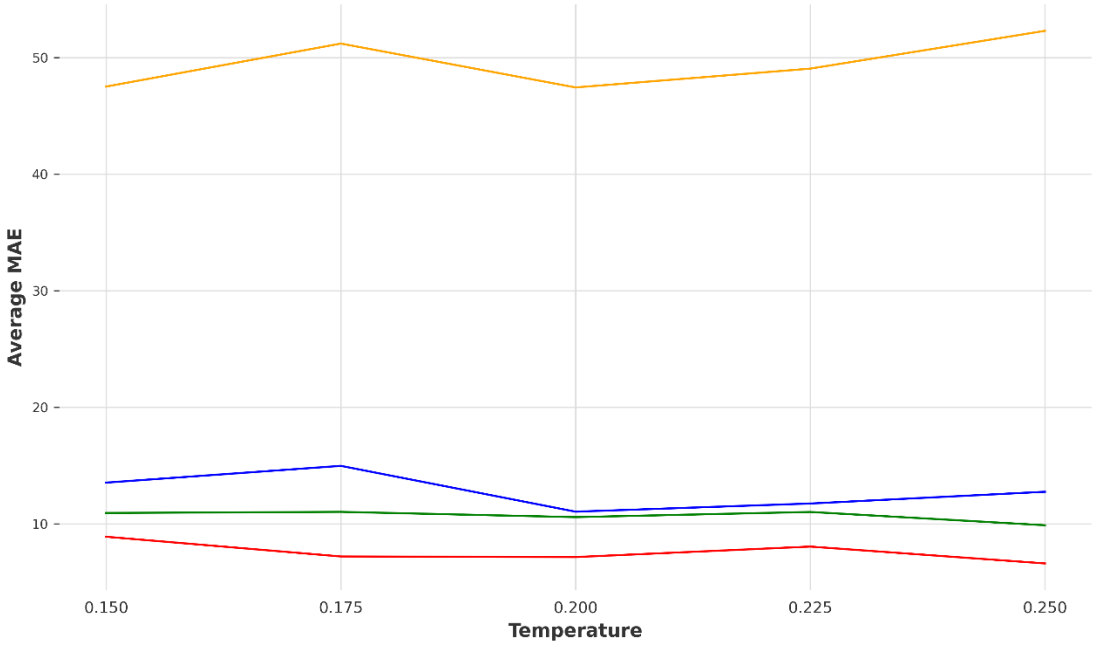


Fig. 3 Ablation study on different temperatures where the blue line represents the AirPassengers data, the red line represents the MonthlyMilk data, the green line represents the AusBeer data, and the orange line represents the Sunspots data

4.3. Application to the stock

For our extended experiment to evaluate forecasting performance, we utilize the Google Stock Price dataset from Kaggle, which provides historical data for Alphabet Inc. (GOOG). The dataset contains daily-level records with the following columns: Date, Open, High, Low, Close, and Volume. The data spans from January 2013 to January 2024. In our experiment, we set the opening price as the target variable for forecasting. To evaluate our model, we use data after June 2023 as the test set. Additionally, we employ a four-horizon window size for performance evaluation, specifically 24, 48, 96 and 120 time steps. We consider Informer, FedFormer and PatchTST as baselines for supervised methods. For zero-shot learning, we compare against LPTM, LLMTime, and LSTPrompt. The experimental results for MAE are presented in Table 6. On this extended dataset, we observe that zero-shot methods generally outperform supervised ones, a trend that is consistent with our earlier observations on the Darts dataset. Despite this trend, our model achieves the best performance across three of the four horizon window sizes, ranking second only on the shortest window of 24. These promising results further strengthen our confidence in the scalability and adaptability of our approach.

Table 6 MAE performance evaluation on the stock dataset. The first-place results are highlighted in bold and the second-place results are underlined

H	Supervised Methods			Zero-Shot Methods			
	Informer	FedFormer	PatchTST	LPTM	LLMTime	LSTPrompt	Ours
24	5.07	8.73	4.52	0.73	0.51	**0.32**	<u>0.42</u>
48	8.03	9.56	4.11	0.80	0.42	<u>0.19</u>	**0.18**
96	3.11	9.43	4.36	0.87	1.42	<u>0.41</u>	**0.33**
120	4.07	10.59	4.65	1.28	2.61	<u>0.52</u>	**0.50**

5. Conclusion and future work

In this paper, we address the task of TSF by introducing a hierarchical collaboration framework that leverages multiple LLM-based agents across multiple layers. Our approach is zero-shot and does not rely on fine-tuning or retrieval-augmented generation. Instead, it harnesses the power of advanced prompting techniques to guide the forecasting process. Applied to both the Darts benchmark and real-world stock datasets, our method demonstrates promising performance and shows the potential of LLMs in time series scenarios without conventional supervision.

While our approach demonstrates some positive results on both Darts and stock datasets, there is still room for further improvement. First, our integration of multi-layer and multi-agent architectures results in the inference latency and computational overhead of large LLMs, posing significant limitations for time-sensitive tasks. Future research could explore hierarchical compression strategies and asynchronous agent coordination to reduce latency while preserving the benefits of multi-agent reasoning. Second, despite advancements in context length, LLMs still struggle with long-range dependencies in TSF. Enhancing their ability to process extended historical sequences may require techniques such as chunking and memory modules. Finally, we plan to further extend our research by applying our proposed solution to a broader range of time series tasks, such as anomaly detection, in order to evaluate its generalizability and effectiveness beyond forecasting.

Declarations

Funding Not applicable.
Conflicts of interest/Competing interests The authors declare that they have no competing interests.
Ethics approval and consent to participate Not applicable.
Availability of data and material Not applicable.
Consent for publication Not applicable.

Authors' contributions CN Tsai designed the method and implemented the code. J Xie conducted the method investigation and contributed to the manuscript review. CM Lai participated in the review of the manuscript. CS Lin supervised the research and contributed to the writing of the manuscript.
Acknowledgement Not applicable.
Clinical trial number Not applicable.

References

1. Zhang, Z., Amiri, H., Liu, Z., Zhao, L., & Züfle, A. (2024, October). Large language models for spatial trajectory patterns mining. In Proceedings of the 1st ACM SIGSPATIAL International Workshop on Geospatial Anomaly Detection (pp. 52-55).
2. Tan, M., Merrill, M., Gupta, V., Althoff, T., & Hartvigsen, T. (2025). Are language models actually useful for time series forecasting?. Advances in Neural Information Processing Systems, 37, 60162-60191.
3. Wang, X., Feng, M., Qiu, J., Gu, J., & Zhao, J. (2025). From news to forecast: Integrating event analysis in llm-based time series forecasting with reflection. Advances in Neural Information Processing Systems, 37, 58118-58153.
4. Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., & Muller, P. A. (2019). Deep learning for time series classification: a review. Data mining and knowledge discovery, 33(4), 917-963.
5. Lim, B., & Zohren, S. (2021). Time-series forecasting with deep learning: a survey. Philosophical Transactions of the Royal Society A, 379(2194), 20200209.
6. Ye, J., Zhang, W., Yi, K., Yu, Y., Li, Z., Li, J., & Tsung, F. (2024). A survey of time series foundation models: Generalizing time series representation with large language model. arXiv preprint arXiv:2405.02358.
7. Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., ... & Liang, P. (2021). On the opportunities and risks of foundation models. arXiv preprint arXiv:2108.07258.
8. Thorp, H. H. (2023). ChatGPT is fun, but not an author. Science, 379(6630), 313-313.
9. Xue, H., & Salim, F. D. (2023). Promptcast: A new prompt-based learning paradigm for time series forecasting. IEEE Transactions on Knowledge and Data Engineering, 36(11), 6851-6864.
10. Chang, C., Wang, W. Y., Peng, W. C., Chen, T. F., & Samtani, S. (2024). Align and Fine-Tune: Enhancing LLMs for Time-Series Forecasting. In NeurIPS Workshop on Time Series in the Age of Large Models.
11. Gruver, N., Finzi, M., Qiu, S., & Wilson, A. G. (2023). Large language models are zero-shot time series forecasters. Advances in Neural Information Processing Systems, 36, 19622-19635.
12. Liu, H., Zhao, Z., Wang, J., Kamarthi, H., & Prakash, B. A. (2024). Lstprompt: Large language models as zero-shot time series forecasters by long-short-term prompting. arXiv preprint arXiv:2402.16132.
13. Zhou, Z., & Yu, R. (2024). Can LLMs Understand Time Series Anomalies?. arXiv preprint arXiv:2410.05440.
14. Tseng, Y. M., Huang, Y. C., Hsiao, T. Y., Chen, W. L., Huang, C. W., Meng, Y., & Chen, Y. N. (2024). Two tales of persona in llms: A survey of role-playing and personalization. arXiv preprint arXiv:2406.01171.
15. Tran, K. T., Dao, D., Nguyen, M. D., Pham, Q. V., O'Sullivan, B., & Nguyen, H. D. (2025). Multi-Agent Collaboration Mechanisms: A Survey of LLMs. arXiv preprint arXiv:2501.06322.
16. Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. Neurocomputing, 50, 159-175.
17. Williams, C., & Rasmussen, C. (1995). Gaussian processes for regression. Advances in neural information processing systems, 8.
18. Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2020). DeepAR: Probabilistic forecasting with autoregressive recurrent networks. International journal of forecasting, 36(3), 1181-1191.
19. Bai, S., Kolter, J. Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv preprint arXiv:1803.01271.
20. Oreshkin, B. N., Dudek, G., Pełka, P., & Turkina, E. (2021). N-BEATS neural network for mid-term electricity load forecasting. Applied Energy, 293, 116918.
21. Zeng, A., Chen, M., Zhang, L., & Xu, Q. (2023, June). Are transformers effective for time series forecasting?. In Proceedings of the AAAI conference on artificial intelligence (Vol. 37, No. 9, pp. 11121-11128).
22. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.
23. Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., & Zhang, W. (2021, May). Informer: Beyond efficient transformer for long sequence time-series forecasting. In Proceedings of the AAAI conference on artificial intelligence (Vol. 35, No. 12, pp. 11106-11115).
24. Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., & Jin, R. (2022, June). Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In International conference on machine learning (pp. 27268-27286). PMLR.
25. Nie, Y., Nguyen, N. H., Sinthong, P., & Kalagnanam, J. (2022). A time series is worth 64 words: Long-term forecasting with transformers. arXiv preprint arXiv:2211.14730.
26. Prabhakar Kamarthi, H., & Prakash, B. A. (2024). Large Pre-trained time series models for cross-domain Time series analysis tasks. Advances in Neural Information Processing Systems, 37, 56190-56214.

27. Das, A., Kong, W., Sen, R., & Zhou, Y. (2024, July). A decoder-only foundation model for time-series forecasting. In Forty-first International Conference on Machine Learning.
28. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., ... & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. Advances in neural information processing systems, 35, 24824-24837.
29. Yang, C., Wang, X., Lu, Y., Liu, H., Le, Q. V., Zhou, D., & Chen, X. (2023). Large language models as optimizers. arXiv preprint arXiv:2309.03409.
30. Herzen, J., Lässig, F., Piazzetta, S. G., Neuer, T., Tafti, L., Raille, G., ... & Grosch, G. (2022). Darts: User-friendly modern machine learning for time series. Journal of Machine Learning Research, 23(124), 1-6.