
Adaptive Information Routing for Multi-Modal Time Series Forecasting

Jun Seo Hyeokjun Choe Seohui Bae Soyeon Park
Jinseok Yang Dongwan Kang Woohyung Lim

LG AI Research

Seoul, Republic of Korea

{jun.seo, hyeokjun.choe, seohui.bae, julia.park,
jinseok.yang, evan.kang, w.lim}@lgresearch.ai

Abstract

Time series forecasting is a critical task for artificial intelligence with numerous real-world applications. Traditional approaches primarily rely on historical time series data to predict the future values. However, in practical scenarios, this is often insufficient for accurate predictions due to the limited information available. To address this challenge, multimodal time series forecasting methods which incorporate additional data modalities, mainly text data, alongside time series data have been explored. In this work, we introduce the Adaptive Information Routing (AIR) framework, a novel approach for multimodal time series forecasting. Unlike existing methods that treat text data as an auxiliary input for forecasting, AIR leverages text information to dynamically guide the time series model by controlling how and to what extent multivariate time series information should be combined. Experiment results demonstrate that AIR effectively modulates the behavior of the time series model using textual inputs, significantly enhancing forecasting accuracy in real-world forecasting tasks. Using synthetic data, we demonstrate that AIR can adjust the integration of time series information in time series forecasting model based on textual cues. Furthermore, experiments with stock price data confirm that AIR enhances the performance of time series forecasting model by effectively leveraging text information.

1 Introduction

Time series forecasting is a crucial objective in artificial intelligence with a wide range of valuable applications. To tackle this problem, abundant forecasting models have been developed [1, 2, 3, 4, 5, 6, 7, 8, 9]. Existing methods solely rely on time series data, neglecting other modalities of information, whereas human experts typically draw insights from diverse data sources. Inspired by the success of the Large Language Models (LLMs), recent research has begun exploring the integration of text data alongside time series data for improved forecasting. Several efforts have focused on fine-tuning LLMs for time series forecasting to leverage the LLMs’ ability to model sequential dependencies [10, 11, 12, 13, 14]. Some researches in this line of approaches [13, 14] attempt to fuse text and time series data by inputting patch embeddings of time series along with word embeddings of text into LLM-based forecasting models. These models capitalize on LLMs’ capacity to process both modalities, potentially enhancing forecast accuracy. Other researches take a different approach by transforming time series data to resemble text-like structures, enabling LLMs to directly handle it and exploit their inherent general knowledge [15, 16, 17]. These approaches demonstrate improved forecasting performance when using prompts containing appropriate text information [17]. Beyond relying on the capabilities of LLMs, there have also been attempts to design multimodal forecasting model itself [18, 19]. Rather than using raw text data, these models typically employ

LLM-derived text embeddings and combine time series and text information using mechanisms such as cross-attention [18] or by integrating separate forecasting outputs from both modalities [19]. This line of research aims to improve forecasting by explicitly modeling the interaction between time series and textual data.

While prior works have shown remarkable achievements in multimodal time series forecasting, these methods typically treat text data as auxiliary inputs to the forecasting model, without adequately addressing the fundamental differences between these modalities. Time series data is typically local, specific, and structured, whereas text data often provides more global, ambiguous, and unstructured information. To account for this distinction, we propose a novel approach that leverages text information as a controller to guide the processing of time series data within the forecasting model. By doing so, the text information can influence the entire forecasting process, rather than providing information for a specific point in time and impacting only isolated moments. Our focus is on multivariate time series forecasting, where the operation of model involves combining information from multiple features into high-level representations. By controlling how and to what extent this feature mixing occurs, we can effectively control the work of the forecasting model. Under this motivation, we introduce the Adaptive Information Routing (AIR) framework, a novel multimodal time series forecasting method that dynamically adjusts the information pathways along which time series information flows within the model based on textual cues.

2 Method

In this section, we present our Adaptive Information Routing (AIR) framework, designed to modulate the behavior of multivariate time series forecasting models by adjusting the information pathways within the model based on textual information. To illustrate our approach, we begin with the fully connected (FC) layer, a basic component of neural networks. FC layer connects each input node to each output node, with the connection strengths represented by the weights. To enable control of the operation of FC layer, we decompose the FC layer into two distinct FC layers with an intermediate latent representation. Since each latent node is connected to all input and output nodes with distinct weights, each nodes in the latent represents a specific connection between input and output nodes. Therefore, we can adjust information pathways between input and output in FC layer by decomposing FC layer into two FC layers and weighting the latent nodes between them. AIR framework leverages this decomposition structure and generates the weights applied to the latent nodes from the text embeddings. By adjusting the connections between input and output nodes adaptively to the text information, AIR guides the behavior of time series model based on the text information.

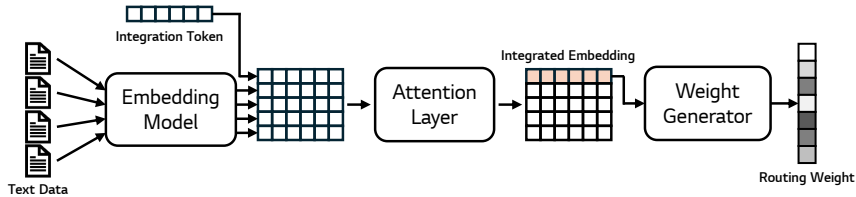


Figure 1: Architecture of information routing module.

Figure 1 illustrates the architecture of the information routing module within the AIR framework. This module comprises three key components: the embedding model, the embedding integrator, and the weight generator. First, each text data is embedded into a fixed-size vector via embedding model. For the embedding model, we adopt Mistral-7B [20] model fine-tuned with E5 [21]. Then, the embedding integrator integrates the information from multiple text embeddings. The embedding integrator consists of an attention layer and a learnable integration token having the same size with text embeddings. The text embeddings and the integration token are fed into the attention layer, and the output associated with the integration token is used as the integrated embedding of the text data. Subsequently, the weight generator produces weights for the latent nodes based on the integrated embedding. We implement the weight generator using a simple Multi-Layer Perceptron (MLP) architecture, applying a softmax activation function to the generated weights to facilitate a weighted combination across multiple paths. To enhance performance further, we group the latent nodes into several clusters and perform softmax normalization within each group.

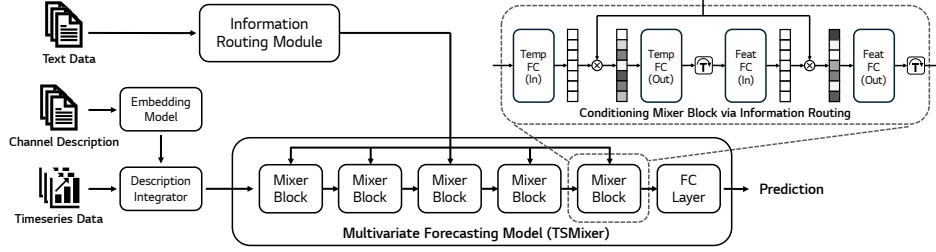


Figure 2: Adaptive Information Routing framework applied to TSMixer.

3 Experiment Results

In this section, we demonstrate the experiment results on AIR framework. We use TSMixer [8], a multivariate forecasting model based on MLPmixer [22], as our base model for integration with AIR. Since TSMixer has a simple architecture consisting of only FC layers, TSMixer is not biased toward any particular type of pattern or feature and possesses minimal inductive bias. Therefore, TSMixer serves as a good testbed for our approach to dynamically adjusting the information pathways within the time series forecasting model. Figure 2 illustrates the architecture of TSMixer integrated with the AIR framework. To incorporate domain-specific information into the routing process, we embed the descriptions of channels in time series data and integrate description embeddings with corresponding time series data, by concatenating the time series and description embedding and passing through an FC layer. For each FC layer in the mixer block, we generate the latent routing weights using information routing module. Note that the information routing is not applied to the final predictor.

3.1 Dataset

For the dataset, we utilize two distinct dataset; a synthetic dataset to verify the effectiveness of AIR framework in dynamically changing forecasting environment, and a stock price dataset to demonstrate the ability of AIR framework to enhance forecasting performance in real-world scenarios.

Synthetic Dataset We generate the synthetic dataset using Timesynth [23] package. For the synthetic data, we set both the input window length and the forecast horizon length to $L = 100$. First, we generate $N_s = 30$ base time series using a random generator. Since these base time series lacks trend, periodicity, or other temporal characteristics, it is challenging to predict them based solely on historical data. Next, we generate a target time series by choosing $N_t = 2$ time series from base time series, and lagging them by the L , and averaging them. Since the forecasting length matches the lagging length, the target time series can be easily predicted by simply averaging appropriate N_t base time series from the input window, if the model correctly identifies the target configuration. To simulate a dynamically changing environment, we alter the target time series configuration every $P = 300$ timestamps, with a small overlapping period of $P_o = 20$. During the overlap, the target time series transitions linearly from the former configuration to the new configuration, with the proportion of the latter configuration increasing gradually. For the text data, we generate documents every P_o timestamps, detailing the target configuration information for timestamps after L timestamps. During the overlapping period, documents describe the transition from the former to the new target configuration. Additional details about the synthetic dataset can be found in the Appendix.

Stock Price Dataset To assess the effectiveness of the proposed AIR framework in real-world forecasting scenarios, we use a stock price dataset. The objective of stock price forecasting experiment is to forecast closing price of the target stock based on the historical closing prices of $N_{stock} = 84$ base stocks. Both the input length and prediction horizon are set to 20 days (4 weeks). Given the abundance of news and reports on the stock market, acquiring and processing relevant text data can be challenging. To address this, we leverage documents generated by a large language model (LLM). Drawing inspiration from TEMPO [13], we utilize the LLM to summarize news and reports published over a specific week regarding the target stock. Specifically, we use ChatGPT-4-turbo [24] to generate text data that includes information about significant events related to the target company, such as cooperations and conflicts with other companies. We assume that the generated summary document is published at Friday of the week, and input document with closing price of Friday. In the experiment, we focus on forecasting the closing prices of Magnificent 7 (M7) stocks. We utilize the closing price

data spanning from 2013 to 2022 as the time series data. The AIR framework is evaluated through a weekly training experiment, where the model is updated every week with data collected up to the previous week, and predictions are made and evaluated daily within that week. The experiment starts from the first week of 2020, and we evaluate the average performance over a three-year period until 2022. Further details regarding the prompts and dataset are provided in the Appendix.

3.2 Results

Table 1 presents the experimental results on the synthetic dataset. We evaluate performance using the mean-squared error (MSE), averaged across three different random seeds. We compare the AIR framework with two baseline methods that incorporate text data into the forecasting model; a method that integrates text embeddings as additional channels within a multivariate forecasting model (Baseline1) and a method that utilizes a separate forecasting network that uses text embeddings to produce predictions, which are then combined with the forecasts from the multivariate model (Baseline2). For Baseline2, we utilize the TSMixer architecture for the text-based forecasting component. The results reveal that both TSMixer and the baseline methods struggle to adapt to the dynamically changing forecasting environment. In contrast, the AIR framework combined with TSMixer effectively adapts to these changes, resulting in significantly lower MSE. This demonstrates that the AIR framework can successfully control the behavior of the multivariate forecasting model based on text information.

Model	MSE
TSMixer	0.4711
TSMixer + Baseline1	0.4668
TSMixer + Baseline2	0.4668
TSMixer + AIR	0.0751

Table 1: Synthetic data experiment results.

Table 2 displays the results of the stock price forecasting experiment. We evaluate performance using the mean-squared error (MSE), averaged across three different random seeds. Among the baseline methods, Baseline1 demonstrates small performance improvements over TSMixer for three stocks (AAPL, META, NVDA), while Baseline2 shows no performance gains. In contrast, the proposed AIR framework achieves substantial performance improvements across all stocks. Specifically, AIR reduces the loss by up to 23.68% for AAPL and by at least 13.60% for MSFT. On average, AIR achieves a 16.93% reduction in loss across all M7 stocks. These results highlight that adaptively adjusting the information path in a multivariate forecasting model based on text data can lead to significant enhancements in forecasting performance.

Model	AAPL	AMZN	GOOGL	META	MSFT	NVDA	TSLA
TSMixer	0.1862	0.0920	0.1307	0.1390	0.0875	0.2730	2.1158
TSMixer + Baseline1	0.1796	0.0931	0.1334	0.1339	0.0889	0.2531	2.1343
TSMixer + Baseline2	0.2069	0.0997	0.1395	0.1433	0.0947	0.2752	2.4359
TSMixer + AIR	0.1421	0.0803	0.1102	0.1193	0.0756	0.2257	1.6648

Table 2: M7 stock price forecasting experiment results.

4 Conclusion

In this paper, we introduce the Adaptive Information Routing (AIR) framework, a novel approach for multimodal time series forecasting. AIR framework modifies the behavior of multivariate forecasting model adaptively to the text data by adjusting connection between input and output nodes in fully-connected layers. By decomposing FC layer into two distinct FC layers with an intervening latent layer, AIR can adaptively control the information path through which the time series information flows by weighting the latent between the decomposed FC layers. Experiment results with synthetic data show that the proposed AIR framework can actually control the dependencies between multivariate time series data based on the text data. Furthermore, with the experiments using stock price data, we show that the proposed AIR framework is able to utilize the text information to improve the forecasting performance in real world time series forecasting problem.

References

- [1] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, “Informer: Beyond efficient transformer for long sequence time-series forecasting,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 12, 2021, pp. 11 106–11 115.
- [2] H. Wu, J. Xu, J. Wang, and M. Long, “Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 22 419–22 430, 2021.
- [3] A. Zeng, M. Chen, L. Zhang, and Q. Xu, “Are transformers effective for time series forecasting?” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 37, no. 9, 2023, pp. 11 121–11 128.
- [4] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, “A time series is worth 64 words: Long-term forecasting with transformers,” in *The Eleventh International Conference on Learning Representations*, 2023.
- [5] B. N. Oreshkin, D. Carпов, N. Chapados, and Y. Bengio, “N-beats: Neural basis expansion analysis for interpretable time series forecasting,” in *International Conference on Learning Representations*, 2020.
- [6] C. Challu, K. G. Olivares, B. N. Oreshkin, F. Garza, M. Mergenthaler, and A. Dubrawski, “N-hits: Neural hierarchical interpolation for time series forecasting,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.
- [7] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long, “Timesnet: Temporal 2d-variation modeling for general time series analysis,” in *The Eleventh International Conference on Learning Representations*, 2023.
- [8] S.-A. Chen, C.-L. Li, S. O. Arik, N. C. Yoder, and T. Pfister, “Tsmixer: An all-mlp architecture for time series forecasting,” *Transactions on Machine Learning Research*, 2023.
- [9] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, and M. Long, “itransformer: Inverted transformers are effective for time series forecasting,” in *The Twelfth International Conference on Learning Representations*, 2024.
- [10] T. Zhou, P. Niu, L. Sun, R. Jin *et al.*, “One fits all: Power general time series analysis by pretrained lm,” *Advances in neural information processing systems*, vol. 36, pp. 43 322–43 355, 2023.
- [11] Y. Bian, X. Ju, J. Li, Z. Xu, D. Cheng, and Q. Xu, “Multi-patch prediction: Adapting llms for time series representation learning,” in *Forty-first International Conference on Machine Learning*, 2024.
- [12] Z. Pan, Y. Jiang, S. Garg, A. Schneider, Y. Nevmyvaka, and D. Song, “@ ip-llm: Semantic space informed prompt learning with llm for time series forecasting,” in *Forty-first International Conference on Machine Learning*, 2024.
- [13] D. Cao, F. Jia, S. O. Arik, T. Pfister, Y. Zheng, W. Ye, and Y. Liu, “Tempo: Prompt-based generative pre-trained transformer for time series forecasting,” in *The Twelfth International Conference on Learning Representations*, 2024.
- [14] X. Liu, J. Hu, Y. Li, S. Diao, Y. Liang, B. Hooi, and R. Zimmermann, “Unitime: A language-empowered unified model for cross-domain time series forecasting,” in *Proceedings of the ACM on Web Conference 2024*, 2024, pp. 4095–4106.
- [15] N. Gruver, M. Finzi, S. Qiu, and A. G. Wilson, “Large language models are zero-shot time series forecasters,” *Advances in Neural Information Processing Systems*, vol. 36, 2024.
- [16] C. Sun, H. Li, Y. Li, and S. Hong, “Test: Text prototype aligned embedding to activate llm’s ability for time series,” in *The Twelfth International Conference on Learning Representations*, 2024.
- [17] M. Jin, S. Wang, L. Ma, Z. Chu, J. Y. Zhang, X. Shi, P.-Y. Chen, Y. Liang, Y.-F. Li, S. Pan *et al.*, “Time-llm: Time series forecasting by reprogramming large language models,” in *The Twelfth International Conference on Learning Representations*, 2024.
- [18] Z. Xu, Y. Bian, J. Zhong, X. Wen, and Q. Xu, “Beyond trend and periodicity: Guiding time series forecasting with textual cues,” *arXiv preprint arXiv:2405.13522*, 2024.

- [19] H. Liu, S. Xu, Z. Zhao, L. Kong, H. Kamarthi, A. B. Sasanur, M. Sharma, J. Cui, Q. Wen, C. Zhang *et al.*, “Time-mmd: A new multi-domain multimodal dataset for time series analysis,” *arXiv preprint arXiv:2406.08627*, 2024.
- [20] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier *et al.*, “Mistral 7b,” *arXiv preprint arXiv:2310.06825*, 2023.
- [21] L. Wang, N. Yang, X. Huang, L. Yang, R. Majumder, and F. Wei, “Improving text embeddings with large language models,” *arXiv preprint arXiv:2401.00368*, 2023.
- [22] I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit *et al.*, “Mlp-mixer: An all-mlp architecture for vision,” *Advances in neural information processing systems*, vol. 34, pp. 24 261–24 272, 2021.
- [23] J. R. Maat, A. Malali, and P. Protopapas, “Timesynth: A multipurpose library for synthetic time series in python,” 2017. [Online]. Available: <http://github.com/TimeSynth/TimeSynth>
- [24] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, “Gpt-4 technical report,” *arXiv preprint arXiv:2303.08774*, 2023.

A Dataset Details

In this section, we provide the details about the datasets we utilize for evaluation in the main paper.

A.1 Synthetic Data

For the synthetic data, we generate the base time series with *PseudoPeriodic*, where the frequency is randomly sampled from a uniform distribution in the range $[0.001, 0.01]$. The frequency variation is set with a standard deviation of 0.01, and the amplitude has a standard deviation of 0.1. Gaussian random noise with a standard deviation of 0.05 is then added. Due to the fact that the frequency deviation exceeds the base frequency and white noise is introduced, the resulting time series exhibit no clear temporal patterns.

Next, we create the target time series by selecting $N_t = 2$ random base series, applying a time lag equal to the forecast horizon length $L = 100$, and averaging the selected series. Figure 3, illustrates the construction process of the target series and the accompanying document data within the synthetic dataset. The document is generated in the form: "The target time series consists of the average of [SERIES A] and [SERIES B]," where [SERIES A] and [SERIES B] correspond to descriptions of two randomly chosen base time series.

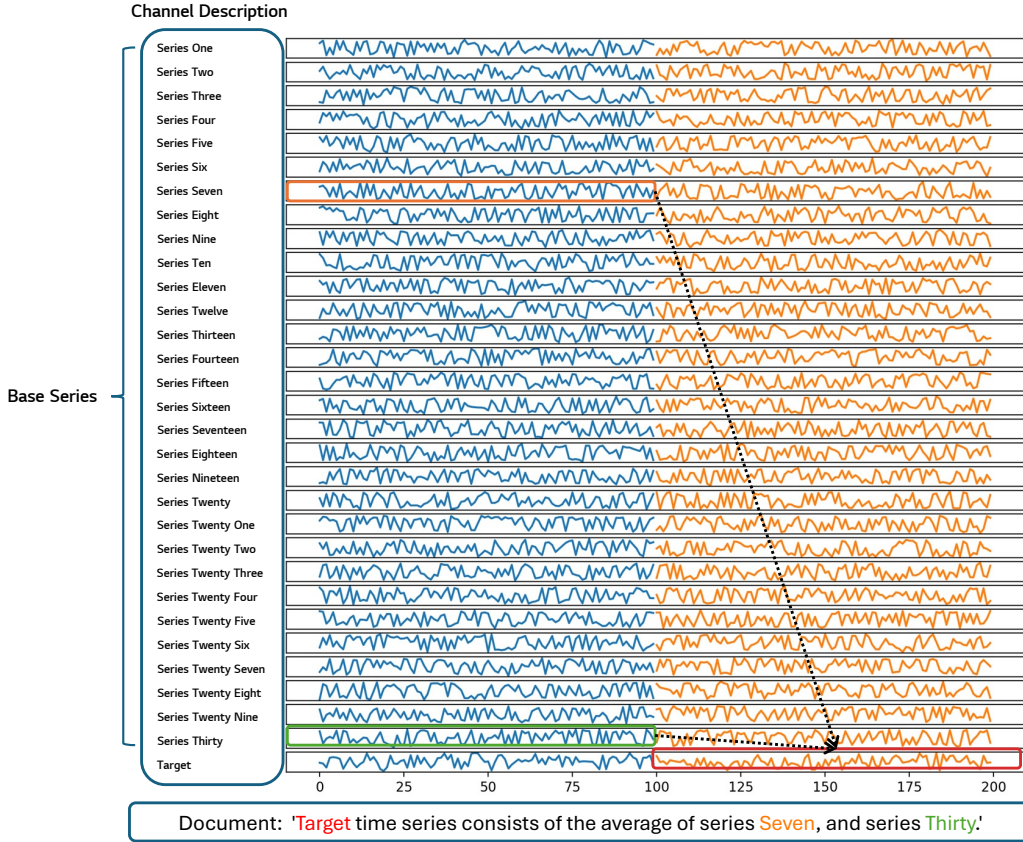


Figure 3: Visualization of Synthetic Time Series and Document Generation.

To simulate a dynamically evolving forecasting environment, we update the target series configuration every $P = 300$ timestamps, with a small overlap of $P_o = 20$ timestamps. During this overlap period, text data is generated in the form: "The configuration of the target series changes from the average of [SERIES A] and [SERIES B] to the average of [SERIES C] and [SERIES D]," where [SERIES A] and [SERIES B] refer to the base time series defining the earlier target configuration, and [SERIES C] and [SERIES D] define the subsequent configuration.

The document, which contains information about the target configuration after L timestamps, is generated at every P_o timestamps and encoded using the E5-Mistral 7B embedding model. These textual embeddings are incorporated into the model by padding timestamps without documents with zeros. To manage the zero-padded data, we apply masking in the attention layer of the information routing module. Additionally, a learnable positional embedding is introduced to encode temporal information, indicating when a document is generated and when a document contains the information about.

Figure 4 visualizes the construction of the input window using the synthetic time series and corresponding documents. If the start of the output horizon aligns with a document generation timestamp, the document is used within the input window with its original timestamp. Otherwise, the most recent document is placed at the start of the input window to ensure the model receives the appropriate configuration information.

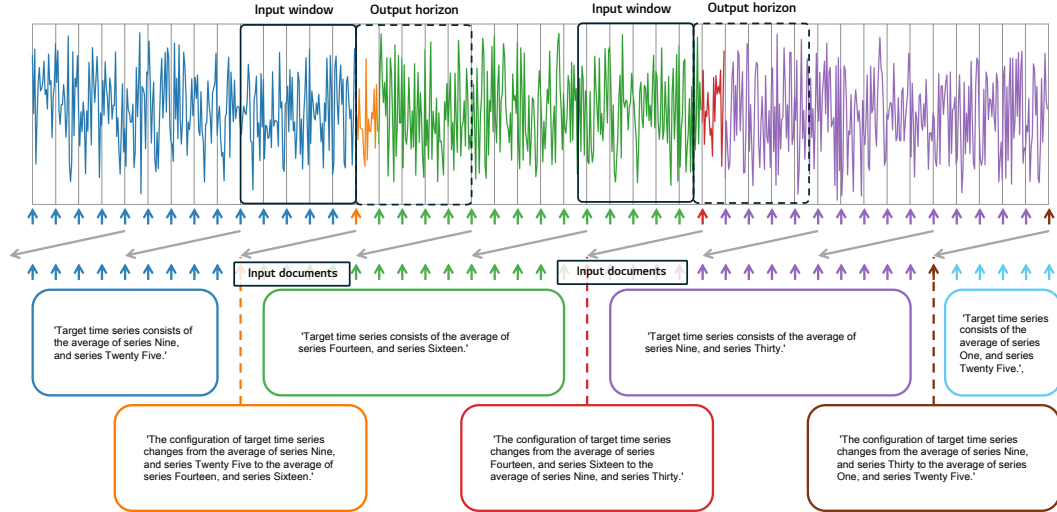


Figure 4: Partitioning Time Series and Document into Window for Synthetic Dataset.

A.2 Stock Price Data

We provide detailed information on the stock price time series data and the prompts used for generating corresponding text data.

For the stock price data, we use the closing prices of stocks within the GICS Information Technology sector that have closing prices available on all trading days from 2013 to 2022 as the base time series. In addition to IT sector stocks, we include stocks closely associated with the information technology sector, such as telecommunications service providers from the Communication Services sector, which includes companies like Alphabet Inc Class A and Meta Platforms Inc. Moreover, we expand the dataset by incorporating stocks from Amazon.com Inc. and Tesla Inc., which belong to the Consumer Discretionary sector. In total, the dataset comprises 84 stocks, with the tickers listed as follows:

AAPL, ACN, ADBE, ADI, ADSK, AKAM, AMAT, AMD, AMZN, ANSS, APH, AVGO, CDNS, CIEN, CRM, CSCO, CSR, CTSH, DIGI, DXC, ENPH, EPAM, FFIV, FICO, FSLR, FTNT, G, GEN, GLW, GOOGL, HPQ, IBM, INTC, INTU, IPGP, IT, JBL, JNPR, KLAC, LRCX, LUMN, MCHP, META, MPWR, MSFT, MSI, MU, NCR, NOW, NTAP, NVDA, NXPI, ON, ORCL, PANW, PSFT, PTC, QCOM, QRVO, ROP, SANM, SMCI, SNPS, SNT, STX, SWKS, T, TDC, TDY, TEL, TER, TMUS, TRMB, TSLA, TXN, TYL, UIS, VIAV, VRSN, VZ, WDC, WU, XRX, ZBRA.

Among these, we designate the "M7" stocks—AAPL, AMZN, GOOGL, META, MSFT, NVDA, and TSLA—as the targets for forecasting.

For each stock, we generate text data on a weekly basis, covering the period from the first week of 2013 to the last week of 2022. Figure 5 illustrates the prompt used to generate the text data, along

with an example corresponding to the 6th week of 2016 for AAPL. We have confirmed that the events mentioned in the GPT-generated response such as financial announcements, collaboration with Cisco, and the conflict with the FBI actually occurred around the specified time frame. Although there may be minor discrepancies in timing and event specifics compared to real-world news, we believe that the generated text accurately reflects significant occurrences, providing sufficient information to support time series forecasting.

To integrate the text data into our model, we embed the generated text into fixed-size vectors using the E5-Mistral 7B embedding model. We assume the summary documents are published at the end of each week, and thus, the text embedding is aligned with the closing price data from the corresponding Friday. For other days of the week, we pad the text embeddings with zeros. During the embedding process, we incorporate positional embeddings to capture temporal context, and apply masking in the attention layer to handle the zero-padded embeddings effectively.

Prompt:

Suppose you are living in [YEAR], can you help me summarize the news and reports in [WEEK] week of [YEAR] for [COMPANY]? If there any news about cooperation or conflict between [COMPANY] and other companies, please include summary about that event with exact company name. Please dedicate each section for summary, cooperation, and conflict without section title. If there is no cooperation or conflict, do not generate corresponding section without any mention. Please directly give me the answer without apology.

Response:

In the 6th week of 2016, Apple Inc. made headlines for several reasons, including ongoing developments and financial updates. The company announced its financial results for the first quarter of the fiscal year 2016, reporting a record quarterly revenue of \$75.9 billion and record quarterly net income of \$18.4 billion. These results were driven by strong sales of iPhone, Apple Watch, and Apple services

Additionally, Apple was reportedly focusing on enhancing its electric vehicle project, known internally as Project Titan. This initiative highlighted Apple's efforts to innovate beyond its traditional product lines, tapping into new technology sectors that could shape future mobility

Cooperation

During this week, Apple Inc. announced a partnership with Cisco Systems aimed at optimizing Cisco networks for iOS devices and apps. This collaboration was designed to deliver a seamless experience for iPhone and iPad users on Cisco corporate networks, enhancing productivity for enterprise users. This move underscored Apple's commitment to strengthening its role in the business sector by providing more integrated solutions for enterprise environments

Conflict

This week also marked an escalation in the ongoing encryption debate between Apple Inc. and the FBI. The conflict centered around the FBI's request for Apple to help unlock an iPhone used by one of the shooters in the December 2015 San Bernardino attack. Apple resisted these demands, citing privacy concerns and the potential to set a precedent that would undermine security for all users. This standoff sparked a broader discussion on encryption, privacy, and national security, drawing attention from various stakeholders across the tech and government sectors.

Figure 5: Prompt and Generated Document Example for Stock Price Dataset.

B Further Details on Adaptive Information Routing

First, we define the multimodal forecasting problem with the mathematical notations. The time series with m channels $\mathbf{X} = \{\mathbf{x}_0, \dots, \mathbf{x}_T\}$, where $\mathbf{x}_t \in \mathbb{R}^m$ is given with the text data $\mathbf{D} = \{\mathbf{d}_0 \dots \mathbf{d}_T\}$. Note that we assume that at most one document is given for each timestamp although multiple documents can exist for a single timestamp in the real world¹. The objective of multimodal time series forecasting is to predict the future values of time series data $\mathbf{X}_{t:t+H} = [\mathbf{x}_t, \dots, \mathbf{x}_{t+H-1}] \in \mathbb{R}^{H \times m}$ using the look-back window $\mathbf{X}_{t-L:t} = [\mathbf{x}_{t-L}, \dots, \mathbf{x}_{t-1}] \in \mathbb{R}^{L \times m}$ and the document history $\mathbf{D}_{t-L:t} = [\mathbf{d}_{t-L}, \dots, \mathbf{d}_{t-1}]$ where L is the look-back window size and H is the length of forecasting horizon. To indicate whether a document exists for a specific timestamp, the mask $\mathbf{M}_{t-L:t} = \{m_{t-L}, \dots, m_t\}$, where $m_t \in [0, 1]$ and 0 indicates document exists for the timestamp, is given with the document data. Furthermore, we utilize the channel description $\mathbf{C} = \{c_0 \dots c_{m-1}\}$ to associate the domain-specific information with the text information in document data. Using large language model fine-tuned as a text embedding model, the document data $\mathbf{D}_{t-L:t}$ and channel description $\mathbf{C}$ are embedded to the embeddings with a fixed size d_{emb} $\mathbf{E}_{t-L:t}^T \in \mathbb{R}^{L \times d_{emb}}$ and $E^C \in \mathbb{R}^{m \times d_{emb}}$.

The information routing model generates the routing weight from document embeddings. Using document embeddings $\mathbf{E}_{t-L:t}^T$ and document mask $\mathbf{M}_{t-L:t}$ and a learnable integration token $\mathbf{T}^{int} \in \mathbb{R}^{d_{emb}}$ and a transformer block *Transformer*, we obtain the integrated embedding $\mathbf{E}^{int} \in \mathbb{R}^{d_{emb}}$.

$$\begin{aligned} \mathbf{E}^{attn} &= \text{Transformer}(\text{concat}(\mathbf{T}^{int}, \mathbf{E}_{t-L:t}^T), \text{concat}(0, \mathbf{M}_{t-L:t})) \\ \mathbf{E}^{int} &= \mathbf{E}^{attn}[0, :], \end{aligned}$$

where *Transformer*($\mathbf{X}, \mathbf{M}$) means a transformer layer with input $\mathbf{X}$ and mask $\mathbf{M}$, $\text{concat}(\mathbf{X}, \mathbf{Y})$ concatenates two matrices along the first dimension. Then, using MLP layers of f_{temp} and f_{feat} with GeLU activation, we obtain features $\mathbf{h}_{temp}$ and $\mathbf{h}_{feat}$ from $\mathbf{E}^{int}$.

$$\mathbf{h}_{temp} = f_{temp}(\mathbf{E}^{int}) \quad \mathbf{h}_{feat} = f_{feat}(\mathbf{E}^{int})$$

where $\mathbf{h}_{temp}, \mathbf{h}_{feat} \in \mathbb{R}^{d_{feat}}$. Using a fully connected generator g_{temp} and g_{feat} , we generate scaling features $\mathbf{h}_{temp}^{scale}$ and $\mathbf{h}_{feat}^{scale}$ from $\mathbf{h}_{temp}$ and $\mathbf{h}_{feat}$.

$$\begin{aligned} \mathbf{h}_{temp}^{scale} &= g_{temp}(\mathbf{h}_{temp}), \quad \mathbf{h}_{temp} \in \mathbb{R}^{N_{layer} \times m \times d_{temp}^{latent}} \\ \mathbf{h}_{feat}^{scale} &= g_{feat}(\mathbf{h}_{feat}), \quad \mathbf{h}_{feat} \in \mathbb{R}^{N_{layer} \times L \times d_{temp}^{latent}} \end{aligned}$$

Then the scaling parameters $\mathbf{S}_{temp}$ and $\mathbf{S}_{feat}$ are obtained via applying groupwise softmax activation across the latent dimension to $\mathbf{h}_{temp}^{scale}$ and $\mathbf{h}_{feat}^{scale}$. The groupwise softmax is done by dividing the latent dimension d_{temp}^{latent} into N_{group} groups each of size $d_{temp}^{latent} / N_{group}$, and applying softmax for each group. Note that the different scaling parameters are generated for each layer. Furthermore, different scaling parameters are applied for different timestamps for feature mixing layer, and for different feature for temporal mixing layer in TSMixer.

Given the $\mathbf{S}_{temp} \in \mathbb{R}^{N_{layer} \times m \times d_{temp}^{latent}}$ and $\mathbf{S}_{feat} \in \mathbb{R}^{N_{layer} \times L \times d_{temp}^{latent}}$, we conduct forecasting with TSMixer. Using the timeseries data $\mathbf{X}_{t-L:t}$ and description embedding $E^C \in \mathbb{R}^{m \times d_{emb}}$, we obtain the combined feature $\mathbf{H}^{(0)}$ using fully connected weight $W_{comb} \in \mathbb{R}^{L \times (L + d_{emb})}$.

$$\mathbf{H}^{(0)} = W_{comb}(\text{concat}(\text{transpose}(E^C), \mathbf{X}_{t-L:t})) \quad \mathbf{H}^{(0)} \in \mathbb{R}^{L \times m}.$$

Then, we process $\mathbf{H}^{(0)}$ using mixer blocks and scaling parameters. Each mixer block contains four FC layers of $FC_{temp_in}^{(l)} \in \mathbb{R}^{d_{temp}^{latent} \times L}$, $FC_{temp_out}^{(l)} \in \mathbb{R}^{L \times d_{temp}^{latent}}$, $FC_{feat_in}^{(l)} \in \mathbb{R}^{d_{feat}^{latent} \times m}$, $FC_{feat_out}^{(l)} \in \mathbb{R}^{m \times d_{feat}^{latent}}$. Given the $\mathbf{H}^{(l)}$, the l th mixer block works as Algorithm 1.

C Visualization for Synthetic Dataset Experiment

In Figure 6, we visualize the forecasting results of TSMixer and TSMixer + AIR from synthetic dataset. From upper part of Figure 6, we can see that TSMixer fails to predict the target time series, since the target configuration periodically and randomly. On the other hand, in the lower part of Figure 6, we can see that the TSMixer + AIR accurately predicts the target despite of the randomly

¹We assume that the multiple documents are summarized by human expert or the large language model.

Algorithm 1 Process of mixer layer with adaptive information routing

Input : $\mathbf{H}^{(l)}, \mathbf{S}_{temp}^{(l)} = \mathbf{S}_{temp}[0, :, :], \mathbf{S}_{feat}^{(l)} = \mathbf{S}_{feat}[0, :, :]$

Output : $\mathbf{H}^{(l+1)}$

- 1: $\mathbf{R} \leftarrow \mathbf{H}^{(l)}$
 - 2: $\mathbf{H} \leftarrow \text{norm}(\mathbf{H}^{(l)})$ $\triangleright$ norm denotes batch normalization
 - 3: $\mathbf{H} \leftarrow FC_{temp_in}^{(l)} \mathbf{H}$
 - 4: $\mathbf{H} \leftarrow \mathbf{S}_{temp}^{(l)} \otimes \mathbf{H}$ $\triangleright \otimes$ denotes element-wise multiplication
 - 5: $\mathbf{H} \leftarrow FC_{temp_out}^{(l)} \mathbf{H}$
 - 6: $\mathbf{H} \leftarrow \text{transpose}(\text{ReLU}(\mathbf{H}) + \mathbf{R})$
 - 7: $\mathbf{R} \leftarrow \mathbf{H}$
 - 8: $\mathbf{H} \leftarrow \text{norm}(\mathbf{H})$ $\triangleright$ norm denotes batch normalization
 - 9: $\mathbf{H} \leftarrow FC_{feat_in}^{(l)} \mathbf{H}$
 - 10: $\mathbf{H} \leftarrow \mathbf{S}_{feat}^{(l)} \otimes \mathbf{H}$ $\triangleright \otimes$ denotes element-wise multiplication
 - 11: $\mathbf{H} \leftarrow FC_{feat_out}^{(l)} \mathbf{H}$
 - 12: $\mathbf{H}^{(l+1)} \leftarrow \text{transpose}(\text{ReLU}(\mathbf{H}) + \mathbf{R})$
-

changing target configuration. The results prove that AIR successfully reflects the change in the target configuration based on the text information, while the original TSMixer without AIR fails to follow the change of target configuration.

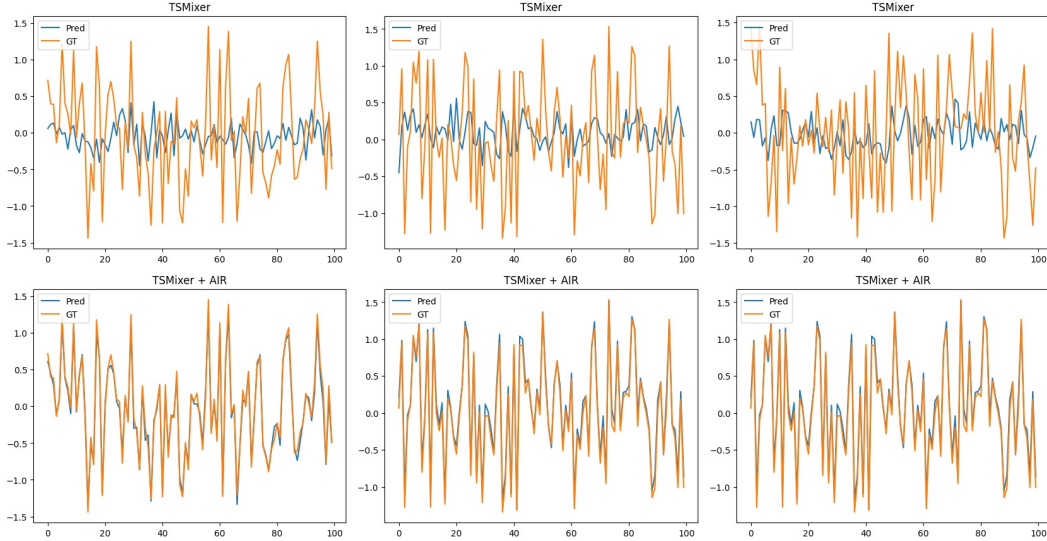


Figure 6: Visualization of Forecasting Result of TSMixer and TSMixer + AIR.