LLM4Time: Uma Ferramenta Interativa para Previsão de Séries Temporais com Modelos Largos de Linguagem

Zairo Bastos[1], **Carlos Caminha**[1,2], **Wellington Franco**[1]

[1]Kunumi Lab - Universidade Federal do Ceará (UFC)
Fortaleza – CE – Brasil

[2]Programa de Pós Graduação em Informática Aplicada - PPGIA/UNIFOR, Brasil

zairo.vianahd@alu.ufc.br,caminha@ufc.br,wellington@crateus.ufc.br

Abstract. *We present LLM4Time, an interactive tool for time series forecasting using Large Language Models (LLMs). The application allows uploading CSV files, automatically handling duplicate values, filling in missing timestamps, and initializing the target column. The system provides predefined prompt engineering strategies such as Zero-Shot, Few-Shot, Chain-of-Thought (CoT), and CoT+Few-Shot (CoT+FS), as well as the ability to connect locally with models via LM Studio, Ollama, and the OpenAI library. A history panel of past forecasts is also available, with filtering options based on the strategy used. The demo aims to facilitate experimentation and analysis of forecasts with LLMs on real-world time series data.*

Resumo. *Apresentamos o LLM4Time*[1] [2]*, uma ferramenta interativa para previsão de séries temporais com Large Language Models (LLMs). A aplicação permite o upload de arquivos CSV, realizando automaticamente a correção de valores duplicados, preenchimento de timestamps ausentes e inicialização da coluna alvo. O sistema oferece estratégias pré-definidas de engenharia de prompts, como Zero-Shot, Few-Shot, Chain-of-Thought (CoT) e CoT+Few-Shot (CoT+FS), além da possibilidade de conexão local com modelos via LM Studio, Ollama e a biblioteca da OpenAI. Também está disponível um painel de histórico das previsões realizadas, com filtragem por estratégia utilizada. A demo tem como foco facilitar a experimentação e análise de previsões com LLMs em dados temporais reais.*

1. Introdução

A previsão de séries temporais é essencial em setores como finanças, saúde, energia e comércio, pois permite identificar padrões e tendências em dados ao longo do tempo. A precisão dessas previsões é crucial para decisões estratégicas, como gestão de estoques e planejamento de recursos [Hyndman and Athanasopoulos 2018].

Tradicionalmente, essa tarefa tem sido abordada por meio de métodos estatísticos, como o modelo *ARIMA* [Mondal et al. 2014], e por técnicas de aprendizagem de máquina, incluindo modelos como *LSTMs* e *Random Forest* [Freitas et al. 2023]. No entanto, essas

[1]Repositório: https://github.com/zairobastos/LLM4Time
[2]Vídeo de demonstração: https://youtu.be/fYSDC_mtjtI

abordagens frequentemente exigem um esforço considerável de engenharia de atributos, ajuste de hiperparâmetros e tratamento minucioso de dados ausentes e sazonalidades.

Nesse contexto, modelos largos de linguagem (LLMs) têm emergido como alternativas promissoras, oferecendo maior flexibilidade, capacidade de adaptação e generalização. Inicialmente projetados para tarefas de processamento de linguagem natural, esses modelos têm demonstrado versatilidade em diversos domínios, como visão computacional [Wang et al. 2024], geração de código [Gu 2023] e, mais recentemente, análise de séries temporais [Jin et al. 2023].

Este trabalho apresenta a ferramenta LLM4Time, que emprega LLMs para a previsão de séries temporais, explorando diferentes estratégias de engenharia de prompts, como *Zero-Shot*, *Few-Shot*, *Chain-of-Thought* e suas combinações. A plataforma dispõe de uma interface interativa que permite ao usuário carregar conjuntos de dados, gerar previsões por meio de APIs (LM Studio, Ollama ou OpenAI) e visualizar métricas de erro. Além disso, os experimentos são armazenados para facilitar comparações e análises detalhadas.

Este artigo está organizado em três seções adicionais. A Seção 2 apresenta os trabalhos relacionados. A Seção 3 descreve a arquitetura da ferramenta LLM4Time, detalhando seu funcionamento e componentes. Por fim, é apresentada às considerações finais e trabalhos futuros.

2. Trabalhos Relacionados

Estudos recentes evidenciam o potencial dos LLMs na previsão de séries temporais, especialmente em abordagens baseadas em engenharia de prompt. [Bastos et al. 2025] demonstraram que o *Gemini-1.5-PRO* superou modelos tradicionais como *LSTMs* e *Random Forest*, mesmo sem treinamento supervisionado, reforçando a eficácia dessas arquiteturas. Abordagens *zero-shot*, como o LLMTIME [Gruver et al. 2023], convertem séries temporais em texto, permitindo que LLMs como *GPT-3* e *LLaMA-2* façam previsões sem *fine-tuning*. De forma complementar, o PromptCast [Xue and Salim 2023] trata a tarefa como uma transformação textual, destacando o uso direto de LLMs com prompts estruturados.

No campo das ferramentas, destaca-se a *Metanalysis* [da Conceição et al. 2020], plataforma interativa voltada a estudantes que integra visualizações e previsões com métodos clássicos. Já soluções comerciais como o *TimeGPT* [Garza and Mergenthaler-Canseco 2023], apesar do alto desempenho, são pagas e pouco transparentes; e bibliotecas como o Darts [Herzen et al. 2022] exigem conhecimento técnico avançado.

Neste contexto, propomos o *LLM4Time*, uma solução intermediária que combina o poder dos LLMs com uma interface gráfica interativa, permitindo a experimentação de estratégias de *prompting*. O objetivo é tornar a previsão com LLMs mais acessível, compreensível e utilizável por um público mais amplo.

3. LLM4Time

A arquitetura do LLM4Time é dividida em quatro módulos — upload de dados, configuração de modelos e prompts, geração de previsões e visualização de histórico — permitindo uma interação flexível e independente em cada etapa, o que facilita a experimentação e a comparação entre diferentes estratégias e modelos.

A ferramenta foi desenvolvida em Python, com interface web implementada em Streamlit[3], proporcionando uma experiência interativa e acessível. Para manipulação e análise dos dados, foram utilizadas as bibliotecas Pandas[4] e NumPy[5], enquanto os gráficos interativos foram construídos com Plotly[6]. Os dados dos experimentos são armazenados localmente em um banco SQLite, e os modelos de linguagem podem ser acessados por meio de APIs da OpenAI[7], LM Studio[8] ou Ollama[9], oferecendo flexibilidade quanto ao backend utilizado. A avaliação das previsões é realizada com métricas como SMAPE, MAE e RMSE, utilizando recursos da biblioteca Scikit-Learn[10].

A seguir, cada módulo da arquitetura do LLM4Time é descrito em detalhe, destacando suas funções e como contribuem para a experiência do usuário. A Figura 1 apresenta um fluxograma que ilustra o fluxo entre as principais etapas da ferramenta.

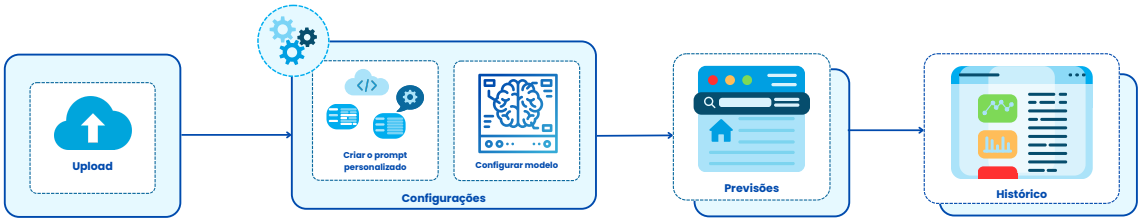

Figura 1. Visão geral da arquitetura da ferramenta LLM4Time, composta pelas etapas de upload dos dados, configuração de prompts e modelos, geração de previsões e histórico de execuções.

3.1. Upload dos Dados e Configurações

O módulo de **Upload dos Dados** é a etapa inicial do fluxo da ferramenta *LLM4Time* e tem como objetivo preparar os dados temporais fornecidos pelo usuário para posterior análise e previsão. Como apresentado na Figura 2, a interface dessa etapa permite o envio de arquivos no formato `CSV`, nos quais o usuário deve identificar duas colunas principais: a coluna que representa o *timestamp* e a coluna contendo as observações a serem previstas.

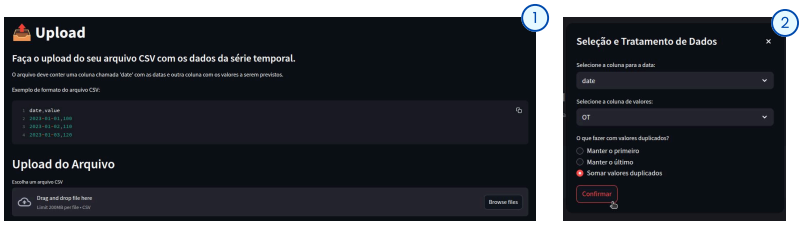

Figura 2. Na etapa inicial, o usuário envia o CSV conforme o padrão solicitado (1) e define as colunas e regras para tratamento de valores duplicados (2).

Após o upload dos dados, o LLM4Time realiza automaticamente um processo de pré-processamento voltado à correção de problemas comuns em séries temporais, como

[3]https://streamlit.io/
[4]https://pandas.pydata.org/
[5]https://numpy.org/
[6]https://plotly.com/
[7]https://pypi.org/project/openai/
[8]https://lmstudio.ai/
[9]https://ollama.com/
[10]https://scikit-learn.org/stable/index.html

valores duplicados e lacunas na sequência temporal. Para duplicatas, o usuário pode optar por remover os registros repetidos ou agregá-los por soma, garantindo um único ponto por instante no tempo. Para valores ausentes, o sistema identifica automaticamente as falhas na sequência de timestamps e oferece três formas estruturadas de preenchimento: atribuição de zero, aplicação de média móvel ou interpolação linear. Esses métodos asseguram a consistência da série e preparam os dados adequadamente para o processo de previsão.

Como apresentado da Figura 3, o módulo de **Configurações** é responsável por centralizar e facilitar o gerenciamento das conexões com modelos de linguagem, além de permitir a criação de versões personalizadas de *prompts*. Essa etapa elimina a necessidade de alterações manuais em código ou arquivos de configuração, oferecendo ao usuário uma interface intuitiva para informar suas credenciais de acesso às APIs.

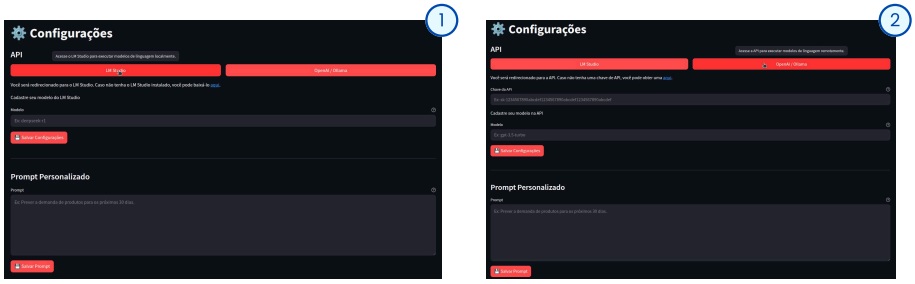

Figura 3. O usuário escolhe a plataforma de execução do modelo (1) e configura a chave da API e cria o prompt personalizado (2).

O módulo de **Configuração** do LLM4Time permite integração com três plataformas para execução de LLMs: LM Studio, Ollama e a biblioteca da OpenAI. O usuário pode escolher a plataforma mais adequada ao seu ambiente, informando apenas as credenciais necessárias diretamente na interface. Além disso, o sistema permite criar e gerenciar prompts personalizados a partir de modelos predefinidos, possibilitando ajustes conforme o contexto de uso, como regras de saída ou formatos específicos de entrada. Essa flexibilidade favorece a experimentação com diferentes estratégias de engenharia de prompts, permitindo testar variações sob o mesmo conjunto de dados de forma prática e acessível.

3.2. Previsões e Histórico

O módulo de **Previsões** é o núcleo da ferramenta *LLM4Time*, onde o usuário realiza experimentações com diferentes modelos, estratégias de *prompting* e configurações de entrada. Como apresentado na Figura 4, a interface permite selecionar o modelo de linguagem desejado (disponível via LM Studio, Ollama ou OpenAI), definir a temperatura da resposta, o intervalo temporal da série a ser enviada ao modelo (data de início e fim) e a quantidade de valores futuros a serem previstos.

Além disso, o usuário pode escolher a estratégia de *prompting* a ser utilizada, dentre as opções pré-definidas: *Zero-Shot*, *Few-Shot*, *Chain-of-Thought (CoT)*, *CoT + Few-Shot (CoT+FS)* ou ainda um *prompt* personalizado criado previamente no módulo de configurações. Também é possível definir o formato de entrada da série a ser enviada ao *prompt*, que pode ser: **Numérico:** os valores são passados diretamente como uma lista de números (ex: 456, 789, 90) e **Textual:** os dígitos de cada valor são separados por

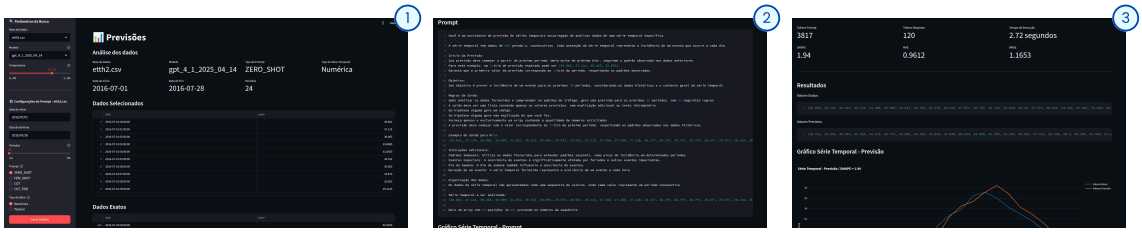

Figura 4. O usuário configura o experimento na barra lateral (1), visualiza o prompt gerado (2) e analisa os resultados da previsão com métricas e gráfico comparativo (3).

espaço em branco, mantendo a separação entre os valores por vírgula (ex: 456, 789, 90 torna-se *"4 5 6, 7 8 9, 9 0"*).

Após a definição de todos os parâmetros, a ferramenta realiza a previsão e apresenta uma interface de resultados rica e interativa. São exibidos:

- A série temporal que foi passada ao modelo;
- O *prompt* final gerado;
- Todos os parâmetros utilizados na execução;
- Um gráfico da série de entrada;
- Os valores reais e previstos, lado a lado;
- As métricas de avaliação da previsão (*SMAPE*, *MAE* e *RMSE*);
- Um gráfico comparativo entre os valores reais e os valores previstos.

Na Figura 5 apresentamos o módulo de **Histórico** do *LLM4Time*. Este módulo exibe todos os testes de previsão já realizados, recuperando os dados salvos no banco da aplicação. A interface permite aplicar filtros por tipo de *prompt* e conjunto de dados utilizado, facilitando a navegação entre os experimentos. Esse módulo é fundamental para análises retrospectivas, pois permite ao usuário comparar estratégias de *prompting*, avaliar o impacto de diferentes configurações (como variações na temperatura) e identificar padrões de desempenho. Com isso, o histórico se torna uma base sólida para aprimorar experimentos futuros e tomar decisões mais precisas no uso dos modelos.

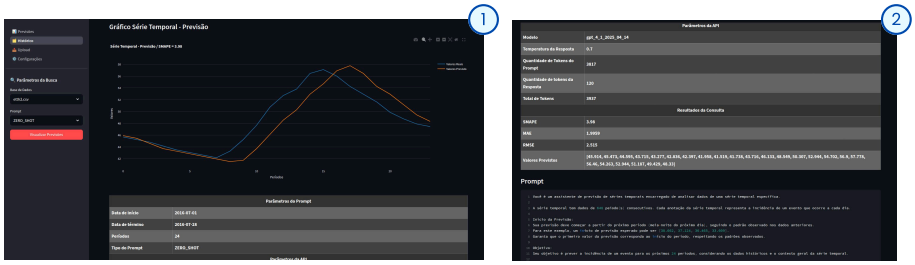

Figura 5. Apresenta o filtro de buscas e os resultados das previsões realizadas (1), além dos detalhes técnicos da consulta, incluindo métricas e prompt utilizado (2)

4. Conclusão e Trabalhos Futuros

Este trabalho apresentou a LLM4Time, uma ferramenta interativa que facilita o uso de LLMs na previsão de séries temporais ao combinar estratégias como *Zero-Shot*,

Few-Shot e *Chain-of-Thought* em uma plataforma unificada. A solução automatiza o pré-processamento, oferece suporte a múltiplos backends de modelos e incentiva a experimentação por meio de métricas padronizadas e registro de resultados. Para o futuro, estão previstos novos recursos, como um módulo de benchmarking para avaliação comparativa entre modelos, empacotamento da aplicação com Docker para simplificar a instalação e melhorias nas etapas de pré-processamento, fortalecendo a LLM4Time como uma ferramenta prática e robusta tanto para pesquisa quanto para aplicações reais.

Referências

Bastos, Z., Freitas, J. D., Franco, J. W., and Caminha, C. (2025). Prompt-driven time series forecasting with large language models. In *Proceedings of the 27th International Conference on Enterprise Information Systems - Volume 1: ICEIS*, pages 309–316. INSTICC, SciTePress.

da Conceição, J. S., dos Santos, J. L., and Cavalcante, R. (2020). Ferramenta para análise de séries temporais. In *Anais da XX Escola Regional de Computação Bahia, Alagoas e Sergipe*, pages 272–281, Porto Alegre, RS, Brasil. SBC.

Freitas, J. D., Ponte, C., Bomfim, R., and Caminha, C. (2023). The impact of window size on univariate time series forecasting using machine learning. In *Symposium on Knowledge Discovery, Mining and Learning (KDMiLe)*, pages 65–72. SBC.

Garza, A. and Mergenthaler-Canseco, M. (2023). Timegpt-1.

Gruver, N., Finzi, M., Qiu, S., and Wilson, A. G. (2023). Large language models are zero-shot time series forecasters. *Advances in Neural Information Processing Systems*, 36:19622–19635.

Gu, Q. (2023). Llm-based code generation method for golang compiler testing. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 2201–2203.

Herzen, J., Lässig, F., Piazzetta, S. G., Neuer, T., Tafti, L., Raille, G., Van Pottelbergh, T., Pasieka, M., Skrodzki, A., Huguenin, N., et al. (2022). Darts: User-friendly modern machine learning for time series. *Journal of Machine Learning Research*, 23(124):1–6.

Hyndman, R. J. and Athanasopoulos, G. (2018). *Forecasting: principles and practice*. OTexts.

Jin, M., Wang, S., Ma, L., Chu, Z., Zhang, J. Y., Shi, X., Chen, P.-Y., Liang, Y., Li, Y.-F., Pan, S., et al. (2023). Time-llm: Time series forecasting by reprogramming large language models. *arXiv preprint arXiv:2310.01728*.

Mondal, P., Shit, L., and Goswami, S. (2014). Study of effectiveness of time series modeling (arima) in forecasting stock prices. *International Journal of Computer Science, Engineering and Applications*, 4(2):13.

Wang, W., Chen, Z., Chen, X., Wu, J., Zhu, X., Zeng, G., Luo, P., Lu, T., Zhou, J., Qiao, Y., et al. (2024). Visionllm: Large language model is also an open-ended decoder for vision-centric tasks. *Advances in Neural Information Processing Systems*, 36.

Xue, H. and 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.