ARAMIS: Uma ferramenta web integrada com LLM open-source para apoio à correção de TCCs de estudantes de graduação

Gustavo Campelo de Sousa[1], Pablo Kauan Martins Timbó[1], Luiz Zairo Bastos Viana[1], Antônio Emerson Barros Tomaz[2], José Wellington Franco da Silva[1], Carlos de Oliveira Caminha[1]
[1]Kunumi Lab – Universidade Federal do Ceará (UFC), Fortaleza – CE
[2]GSIPP – Universidade Federal do Ceará (UFC), Crateús – CE
gustavocraft321@gmail.com, pablokauan@alu.ufc.br, zairobastos@gmail.com, emerson@crateus.ufc.br, wellington@crateus.ufc.br, caminha@ufc.br,

Abstract

A revisão de trabalhos acadêmicos é uma etapa crucial, porém onerosa, na formação de pesquisadores. Trabalhos anteriores obtiveram bons resultados com abordagens automatizadas de revisão em inglês. Nesse contexto, apresentamos o ARAMIS [1], uma ferramenta multiagente baseada em *Large Language Models* (LLM) *open-source*, projetada para revisar Trabalhos de Conclusão de Curso (TCC) em português. A solução foca em três pilares: correção gramatical, encadeamento lógico e rigor metodológico, permitindo ao usuário receber revisões estruturadas para cada pilar escolhido. Mesmo em estágio experimental, os testes atingiram ótimos resultados de usabilidade ao aplicar o *System Usability Scale* (SUS), obtendo uma pontuação de 90,5/100.

1 Introdução

O Trabalho de Conclusão de Curso (TCC) constitui uma etapa importante na formação acadêmica, por representar a consolidação dos conhecimentos desenvolvidos ao longo da graduação e por exigir do estudante domínio técnico, clareza na escrita e organização argumentativa (Guedes and Guedes, 2012). De acordo com a Associação Brasileira de Normas Técnicas (2024), trata-se de um documento elaborado segundo princípios e normas específicas, com a finalidade de obter o grau de bacharelado ou licenciatura. Nesse contexto, a produção do TCC demanda atenção contínua, especialmente porque falhas de escrita e de estrutura comprometem a qualidade do texto e dificultam a comunicação científica (Carboni and Nogueira, 2008). Entre os problemas frequentemente observados em produções acadêmicas, destacam-se erros de grafia, inadequações de pontuação e fragilidades na construção textual (Lunsford and Lunsford, 2008).

Com o avanço das tecnologias de Processamento de Linguagem Natural (PLN), diferentes abordagens de correção automatizada passaram a ser empregadas para apoiar a escrita acadêmica, reduzindo erros tipográficos e gramaticais e melhorando a experiência do usuário (Srivarsha et al., 2025). Mais recentemente, o surgimento dos *Large Language Models* (LLMs) ampliou de forma significativa esse cenário, possibilitando análises mais refinadas e contextualizadas de textos científicos (Liang et al., 2024). Trabalhos recentes têm demonstrado o potencial desses modelos para gerar revisões precisas e direcionadas, sobretudo em língua inglesa, seja com modelos proprietários (Chamoun et al., 2024; D'Arcy et al., 2024), seja com alternativas de código aberto (Idahl and Ahmadi, 2025).

Além das aplicações em linguagem natural, os LLMs vêm demonstrando elevada capacidade de adaptação a diferentes domínios, consolidando-se como uma tecnologia de propósito geral. Esses modelos têm sido utilizados em áreas diversas, como visão computacional (Wang et al., 2024), geração de código (Gu, 2023) e previsão de séries temporais (Bastos et al., 2025b). Esse avanço evidencia que os LLMs podem servir de base para o desenvolvimento de ferramentas especializadas, voltadas não apenas à geração de texto, mas também ao suporte à tomada de decisão, à automação de tarefas e à análise de conteúdos técnicos.

Esse movimento já pode ser observado em aplicações construídas para finalidades específicas. No domínio de séries temporais, por exemplo, soluções como o *LLM4Time* (Bastos et al., 2025a) exploram o uso de LLMs para previsão, enquanto bibliotecas como o *LLM4Series* (Silva et al., 2026) auxiliam na construção de *prompts* e oferecem suporte a diferentes *back-ends* para tarefas preditivas. Em outro contexto, ferramentas como o *OBI-UAN* (Bastos et al., 2025c) utilizam esses modelos para apoiar a explicação de questões de programação. Apesar desse avanço, ainda há uma lacuna no suporte à língua portuguesa, especialmente no que se refere à revisão de textos acadêmicos com foco não apenas

[1]https://youtu.be/lw2GSQ7aiW0

Proceedings of the 17th International Conference on Computational Processing of Portuguese (PROPOR 2026)
Vol. 2, pages 1–4

em aspectos gramaticais, mas também em elementos mais amplos, como coerência argumentativa e rigor metodológico.

Diante dessa lacuna, este trabalho apresenta o ARAMIS (*Academic Review Agents for Methodological Improvements*), uma ferramenta *open-source* voltada à revisão de TCCs em português, fundamentada em Grandes Modelos de Linguagem. A proposta integra um LLM *open-source* a uma arquitetura orientada por *prompts* especializados, capaz de fornecer *feedback* linguístico e estrutural ao estudante. Com isso, busca-se contribuir para a melhoria da qualidade textual e do rigor científico das produções acadêmicas, além de tornar mais eficiente o processo de acompanhamento e orientação.

As seções a seguir apresentam a descrição do sistema, a estratégia de avaliação adotada, os resultados obtidos e as perspectivas de continuidade.

2 Descrição da Ferramenta

O ARAMIS[2] é uma ferramenta *Web* baseada em uma arquitetura multiagente, integrada com o modelo *open-source gpt-oss-20b*[3], pela API do *Hugging Face*. É composta por três agentes: correção gramatical, encadeamento lógico e rigor metodológico. Voltada para alunos de graduação, o ARAMIS tem o intuito de fornecer *feedback* direcionado sobre os elementos do texto inserido. O usuário deve acessar a seção "Nova Correção" da Figura 1 e, nela, preencher algumas informações de pré-configuração disponíveis na interface, além de inserir o texto que receberá a revisão. Uma vez inserido, a geração da revisão inicia-se ao clicar no botão "Iniciar Análise" e, após um curto tempo de espera, o *feedback* é gerado e exibido ao usuário em uma nova seção chamada "Análises", conforme exibido na Figura 2; tal seção exibe para o usuário todas as revisões geradas, permitindo que ele as consulte em qualquer momento oportuno. Outras funcionalidades principais disponíveis são o *login*, o cadastro e a seção de histórico de análises, presentes na seção "Minhas Análises".

2.1 Arquitetura

O ARAMIS é composto por uma arquitetura baseada em multiagentes, na qual cada agente é integrado ao modelo *open-source gpt-oss-20b* e orquestrado pelo *framework Agno*[4]. A Figura 3 exibe o fluxograma da ferramenta, ilustrando, desde a entrada do texto, as etapas de pré-configuração e de preenchimento de *prompts*, a definição da requisição, e segue para a orquestração dos agentes especializados (correção gramatical, encadeamento lógico e rigor metodológico), até a organização das análises e a geração do *review* final consolidado.

Figure 1: Interface da seção "Nova Correção".

Figure 2: Interface da seção "Análises Realizadas".

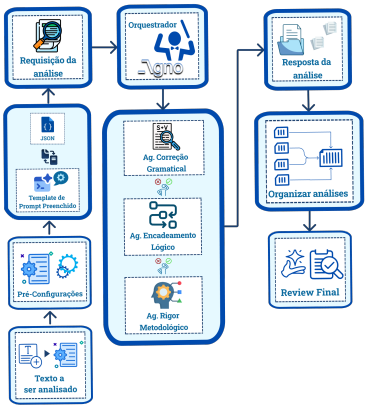

Figure 3: Fluxograma da arquitetura do ARAMIS.

[2] http://enginelab.ufc.br/aramis
[3] https://huggingface.co/openai/gpt-oss-20b
[4] https://docs.agno.com/

2.2 Agentes do ARAMIS

Em sua versão atual, o ARAMIS integra três agentes com características próprias:

- **Correção Gramatical**: Agente voltado para identificar e listar erros ortográficos e equívocos de acentuação;
- **Encadeamento Lógico**: Agente preparado para lidar com a coerência entre sentenças;
- **Rigor Metodológico**: Agente concentrado em verificar a coerência entre o problema de pesquisa, os objetivos, o método e a análise dos resultados.

2.3 Modelagem dos Prompts

Os *prompts* foram estruturados de modo a receberem orientações explícitas sobre os detalhes e o formato da resposta a serem retornados. Além disso, há *placeholders* que o sistema preenche automaticamente com as informações inseridas pelo usuário na etapa anterior à ativação da revisão. Para a modelagem dos *prompts*, utilizou-se o *few-shot prompting* (IBM, 2025). Abaixo, detalhamos quais variáveis são substituídas:

- **secao_desejada**: Indica nos *prompts* a seção que será abrangida para análise;
- **titulo_tcc**: Contém o título do trabalho do aluno;
- **area_conhecimento_tcc**: Representa a área do conhecimento a qual o TCC está atrelado;
- **nivel_rigor_modelo**: Define o nível de rigor com que o modelo deve avaliar o trecho do TCC do aluno.

3 Avaliação e Discussão

A usabilidade do sistema foi avaliada por meio do *System Usability Scale* (SUS) (Brooke, 1995). O SUS é uma métrica que mensura a usabilidade subjetiva, com 10 itens em uma escala *Likert* de 5 pontos, que varia de "Discordo Completamente" (1) a "Concordo Completamente" (5) (Joshi et al., 2015).

O *score* final, conforme a Fórmula 1 do SUS, é calculado pela soma das contribuições individuais de cada item do questionário. Para os itens de numeração ímpar, formulados de maneira positiva, subtrai-se 1 do valor atribuído pelo respondente. Para os itens de numeração par, formulados de maneira negativa, a contribuição é obtida subtraindo-se a resposta de 5.

$$SUS = 2,5 \times \left[\sum_{i=1}^{10} f(x_i) \right] \quad (1)$$

Embora a proposta do SUS não defina categorias qualitativas para a interpretação dos *scores*, este trabalho adota a escala sugerida por Bangor et al. (2009), amplamente utilizada. Essa classificação permite qualificar o desempenho do sistema conforme os intervalos:

- **Excelente**: Pontuação entre 90 e 100;
- **Bom**: Pontuação entre 80 e 89;
- **Aceitável**: Pontuação entre 70 e 79;
- **Precisa Melhorar**: Pontuação entre 60 e 69;
- **Ruim**: Pontuação inferior a 60.

O *feedback* dos usuários indicou a simplicidade de uso, a boa construção das seções, a qualidade e a precisão das revisões dos agentes, especialmente as do agente de correção gramatical, além do bom desempenho do LLM integrado aos agentes, que rapidamente gerou as revisões e identificou corretamente as incoerências, gerando *feedbacks* úteis. Alguns participantes relataram que os comentários retornados pelo LLM são fundamentais para que o usuário possa obter *feedback* sobre o andamento do trabalho e identificar em que precisa melhorar. Por fim, os usuários relataram a intenção de continuar utilizando o ARAMIS para revisar seus trabalhos, sem se limitar aos testes realizados anteriormente.

Apesar dos resultados promissores, este trabalho apresenta limitações relacionadas ao número reduzido de modelos analisados, bem como à quantidade restrita de participantes, composta por dez usuários de uma única instituição, o que pode comprometer a generalização dos resultados. Somam-se a isso as restrições impostas pelo uso de APIs gratuitas, que têm um limite de requisições, além das limitações da infraestrutura de hospedagem utilizada, que por não suportar localmente modelos como o *gpt-oss-20b*, tornou-se necessária a utilização da API externa do *Hugging Face*.

4 Trabalhos Futuros

O sistema proposto está em fase de melhoria contínua, com esforços para aprimorar a usabilidade e a qualidade das revisões geradas pelo modelo *open-source*. Trabalhos futuros pretendem aplicar técnicas comparativas para melhor seleção do modelo integrado à ferramenta. Pretende-se ampliar a base de usuários para realizar a avaliação da usabilidade da ferramenta. Sugere-se o uso de *fine-tuning*

em um modelo *open-source* para maior direcionamento e precisão nas revisões, além de avaliar também a qualidade das revisões geradas. Além disso, planeja-se hospedar o sistema em um servidor que suporte um modelo como o *gpt-oss-20b*, para que não dependa mais de APIs externas.

Agradecimentos

Agradecemos ao Instituto Kunumi pelo financiamento, pela infraestrutura computacional disponibilizada e pelo apoio à realização e apresentação deste trabalho.

Referências

Associação Brasileira de Normas Técnicas. 2024. NBR 14724: Informação e documentação – Trabalhos acadêmicos – Apresentação. Rio de Janeiro: ABNT. 4. ed.

Aaron Bangor, Philip Kortum, and James Miller. 2009. Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of usability studies*, 4(3):114–123.

Zairo Bastos, Carlos Caminha, and Wellington Franco. 2025a. Llm4time: Uma ferramenta interativa para previsão de séries temporais com modelos largos de linguagem. In *Anais Estendidos do XL Simpósio Brasileiro de Bancos de Dados*, pages 100–105, Porto Alegre, RS, Brasil. SBC.

Zairo Bastos, João David Freitas, José Wellington Franco da Silva, and Carlos Caminha. 2025b. Prompt-driven time series forecasting with large language models. In *ICEIS (1)*, pages 309–316.

Zairo Bastos, Raylander Marques, Gabriel Rudan, Marlon Duarte, and Wellington Franco. 2025c. Obi-uan: Um agente para auxiliar nos estudos da olimpíada brasileira de informática. In *Anais Estendidos do XL Simpósio Brasileiro de Bancos de Dados*, pages 94–99, Porto Alegre, RS, Brasil. SBC.

John Brooke. 1995. Sus: A quick and dirty usability scale. *Usability Eval. Ind.*, 189.

Rosadélia Malheiros Carboni and Valnice de Oliveira Nogueira. 2008. Facilidades e dificuldades na elaboração de trabalhos de conclusão de curso. *ConScientiae Saúde*, 3:65–72.

Eric Chamoun, Michael Schlichtkrull, and Andreas Vlachos. 2024. Automated focused feedback generation for scientific writing assistance. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 9742–9763, Bangkok, Thailand. Association for Computational Linguistics.

Mike D'Arcy, Tom Hope, Larry Birnbaum, and Doug Downey. 2024. Marg: Multi-agent review generation for scientific papers. *Preprint*, arXiv:2401.04259.

Qiuhan Gu. 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.

Hermila Tavares Vilar Guedes and Jorge Carvalho Guedes. 2012. Avaliação, pelos estudantes, da atividade" trabalho de conclusão de curso" como integralização do eixo curricular de iniciação à pesquisa científica em um curso de medicina. *Revista Brasileira de Educação Médica*, 36(2):162–171.

IBM. 2025. What is few-shot prompting? https://www.ibm.com/think/topics/few-shot-prompting. Acesso em: 24 ago. 2025.

Maximilian Idahl and Zahra Ahmadi. 2025. OpenReviewer: A specialized large language model for generating critical scientific paper reviews. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (System Demonstrations)*, pages 550–562, Albuquerque, New Mexico. Association for Computational Linguistics.

Ankur Joshi, Saket Kale, Satish Chandel, and D Kumar Pal. 2015. Likert scale: Explored and explained. *British journal of applied science & technology*, 7(4):396.

Weixin Liang, Yuhui Zhang, Hancheng Cao, Binglu Wang, Daisy Yi Ding, Xinyu Yang, Kailas Vodrahalli, Siyu He, Daniel Scott Smith, Yian Yin, and 1 others. 2024. Can large language models provide useful feedback on research papers? a large-scale empirical analysis. *NEJM AI*, 1(8):AIoa2400196.

Andrea A. Lunsford and Karen J. Lunsford. 2008. "mistakes are a fact of life": A national comparative study. *College Composition and Communication*, 59(4):781–806.

Wesley Barbosa Silva, Maria Fernanda Aquino Freitas Scarcela, Luiz Zairo Bastos Viana, Carlos Caminha, João Paulo do Vale Madeiro, and José Wellington Franco da Silva. 2026. LLM4series: Structured prompting for time series forecasting with LLMs. In *1st ICLR Workshop on Time Series in the Age of Large Models*.

Nalla Srivarsha, Gummadi Nithin, Shanmugasundaram Hariharan, Bibhuti Bhusan Dash, Subrata Chowdhury, and Sudhansu Shekhar Patra. 2025. Auto text correction using nlp techniques. In *2025 6th International Conference on Mobile Computing and Sustainable Informatics (ICMCSI)*, pages 590–594. IEEE.

Wenhai Wang, Zhe Chen, Xiaokang Chen, Jiannan Wu, Xizhou Zhu, Gang Zeng, Ping Luo, Tong Lu, Jie Zhou, Yu Qiao, and 1 others. 2024. Visionllm: Large language model is also an open-ended decoder for vision-centric tasks. *Advances in Neural Information Processing Systems*, 36.