ARAMIS: UMA FERRAMENTA MULTIAGENTE INTEGRADA COM LLM OPEN-SOURCE PARA APOIO À CORREÇÃO DE TCCS DE ESTUDANTES DE GRADUAÇÃO

ARAMIS: A MULTI-AGENT TOOL INTEGRATED WITH OPEN-SOURCE LLM FOR SUPPORTING THE CORRECTION OF UNDERGRADUATED STUDENTS THESES

Gustavo Campelo de Sousa [1]
Prof. Dr. José Wellington Franco da Silva (Orientador) [2]
Prof. Dr. Antonio Emerson Barros Tomaz (Coorientador) [3]

RESUMO

A correção do Trabalho de Conclusão de Curso (TCC) é uma etapa crucial na formação da pesquisa do aluno de graduação, no entanto, esse processo pode ser cansativo tanto para o aluno em sua pesquisa quanto para o orientador durante o acompanhamento, devido a fatores como sobrecarga de tarefas e retornos pouco específicos sobre o conteúdo da pesquisa. A automatização na correção de trabalhos científicos, por meio de técnicas como Aprendizado de Máquina (AM) e Processamento de Linguagem Natural (PLN), passou a integrar o cotidiano dos alunos, principalmente após o surgimento dos *Large Language Model*s (LLMs). Neste trabalho, foi desenvolvido o *Academic Review Agents for Methodological Improvements* (ARAMIS), uma ferramenta de análise e correção de TCCs em português, composta por três agentes especializados: correção gramatical, encadeamento lógico e rigor metodológico, e que integra um LLM *open-source* orientado por engenharia de *prompt*. Foi adotada uma abordagem de análise comparativa entre a geração de *feedback* por LLMs proprietários e *open-source*, visando selecionar o modelo que operasse com um *trade-off* satisfatório. A proposta consistiu na integração do melhor modelo de código aberto avaliado ao ARAMIS, desenvolvido no âmbito deste trabalho, focado em fornecer o *feedback* dos TCCs em português analisado, composto por três agentes, pilares da geração da revisão automatizada. A ferramenta recebe o texto do aluno, que é processado pelo LLM, e retorna uma revisão estruturada, baseada nas diretrizes definidas nas configurações dos agentes. Neste trabalho, utilizou-se o modelo *System Usability Scale* (SUS) para avaliar o nível de usabilidade da ferramenta. Os experimentos foram conduzidos com usuários reais em processo de escrita de TCC, no qual o questionário SUS foi aplicado imediatamente após a realização dos testes na ferramenta. Os resultados demonstram que o ARAMIS obteve 90,5/100 pontos, confirmando que a aplicação atende às expectativas de usabilidade dos estudantes de graduação, sendo útil e retornando revisões direcionadas e precisas.

Palavras-chave: Feedback. TCC. LLM. Open-source. ARAMIS. Engenharia de prompt. SUS

[1] Graduando em Ciência da Computação na UFC (Campus Crateús) - gustavocraft@alu.ufc.br
[2] Professor Doutor em Ciência da Computação na UFC (Campus Crateús) - wellington@crateus.ufc.br
[3] Professor Doutor em Ciência da Computação na UFC (Campus Crateús) - emerson@crateus.ufc.br

ABSTRACT

The correction of Undergraduate Final Projects is a crucial stage in the academic development of undergraduate students. However, this process can be time-consuming and exhausting both for students during their research activities and for advisors during supervision, due to factors such as task overload and insufficiently specific feedback on research content. The automation of scientific writing correction using techniques such as Machine Learning (ML) and Natural Language Processing (NLP) has become increasingly present in students' daily routines, especially after the emergence of Large Language Models (LLMs). In this study, Academic Review Agents for Methodological Improvements ARAMIS was developed, a tool for the analysis and correction of undergraduate theses in portuguese, composed of three specialized agents: grammatical correction, logical chaining and methodological rigor, integrating an open-source LLM guided by prompt engineering techniques. A comparative analysis approach was adopted to evaluate the feedback generation by proprietary and open-source LLMs, with the objective of selecting a model that operates with a satisfactory trade-off. The proposed solution consisted of integrating the best-performing evaluated open-source model into ARAMIS, developed within the scope of this study, focused on returning the analyzed undergraduate theses feedback in portuguese, being composed by the three agents, which serve as the core pillars of automated revision generation. The tool receives the student's text, which is processed by the LLM and returns a structured review based on the guidelines defined in the agent's configurations. In this work, the System Usability Scale (SUS) was employed to assess usability. The experiments were conducted with real users actively engaged in undergraduate thesis writing, and the SUS questionnaire was applied immediately after tool usage. The results demonstrate that ARAMIS achieved a score of 90.5/100, confirming that the application meets undergraduate students's usability expectations, being useful and achieving precise and targeted feedback.

Keywords: Feedback. Final Project. LLM. Open-source. ARAMIS. Prompt engineering. SUS

1 INTRODUÇÃO

O Trabalho de Conclusão de Curso (TCC) constitui a etapa final do percurso acadêmico dos estudantes de graduação, sendo um requisito indispensável em boa parte dos cursos para a obtenção do grau de bacharel ou licenciado na respectiva área de formação.

Ao elaborar um texto dessa importância, é necessária a atenção contínua ao trabalho, pois, no processo de avaliação do TCC, observa-se o rigor no desenvolvimento conforme as normas, o encadeamento lógico das ideias, a profundidade e a apresentação do conhecimento do tema estudado (CARBONI; NOGUEIRA, 2008). Diante disso, a revisão textual, seja pelo aluno e/ou orientador, deixou de se limitar apenas a aspectos normativos e passou a observar aspectos como verossimilhança e encadeamento narrativo (PEREZ; BOENAVIDES, 2017). Nesse processo, o discente possui um papel fundamental na garantia da qualidade, atuando em parceria com o orientador — cuja experiência tende a facilitar a identificação de inconsistências textuais. Contudo, mesmo com o empenho conjunto, a necessidade de sucessivas revisões persiste, gerando retrabalho e consumo excessivo de tempo. Diante dessas dificuldades, fica evidente a necessidade de soluções que agilizem a revisão, como sistemas de Inteligência Artificial (IA) projetados para corrigir textos conforme as particularidades do contexto acadêmico.

No estudo de Srivarsha *et al.* (2025), argumenta-se que a correção automatizada aprimora significativamente a experiência do usuário, reduzindo erros tipográficos e linguísticos no conteúdo escrito. Para isso, são incorporados métodos aos computadores para que estes compreendam a linguagem humana, chamados de técnicas de Processamento de Linguagem Natural (PLN) (COPPIN, 2004). Em um estudo anterior, Zhang e Zhang (2023) apresentaram um enfoque mais específico na precisão da tradução automática para o inglês, embora também vise a redução de erros gramaticais, empregando técnicas de análise semântica e métodos especializados de aumento de dados. Tais trabalhos, embora aplicados a domínios ligeiramente diferentes, convergem para o objetivo comum de melhorar a qualidade do texto por meio da detecção e correção automatizada de erros. Tais técnicas aplicadas podem ser adaptáveis à revisão de textos acadêmicos, como os TCCs, visto que em Lunsford e Lunsford (2008) os principais problemas identificados em trabalhos acadêmicos incluem erros de grafia (13,7% dos casos) e a ausência de vírgulas após elementos introdutórios (9,6%), entre outros.

Tais exigências de formalidade gramatical e normativa na elaboração de TCCs e artigos científicos requerem do aluno competências linguísticas bem desenvolvidas. No entanto, segundo a *Organisation for Economic Co-operation and Development* (OECD), no estudo *The State of Learning and Equity in Education*, estudantes brasileiros de 15 anos de idade obtiveram 410 pontos em leitura no *ranking Programme for International Student Assessment* (PISA), abaixo da média da própria OECD, que é de 476 pontos (OECD, 2023). Esse dado revela deficiências na formação básica, especialmente em leitura e interpretação, habilidades fundamentais para a produção textual no ensino superior. Essa deficiência compromete a autonomia dos estudantes para interpretar textos complexos, realizar pesquisas e aplicar o método científico — o que reforça a necessidade de suporte específico no processo de escrita acadêmica.

Embora sistemas de correção automática consigam corrigir a grafia e a pontuação, nenhuma solução de IA foi ainda treinada em português especificamente para diagnosticar e orientar aspectos macroestruturais, como o encadeamento argumentativo de capítulos ou a adequação metodológica de um trabalho de conclusão de curso. Até o momento, essas avaliações permanecem predominantemente a cargo de orientadores humanos, perpetuando ciclos de

retrabalho que atrasam as defesas e reduzem a qualidade científica dos trabalhos. Essa lacuna tecnológica sugere a seguinte questão de pesquisa: *Como sistemas de Inteligência Artificial especializados em textos acadêmicos em português podem diagnosticar e orientar a construção de encadeamentos lógicos e rigor metodológico em Trabalhos de Conclusão de Curso, reduzindo o número de revisões manuais sem comprometer a autoria e a qualidade científica?*

Este trabalho tem como objetivo desenvolver uma ferramenta automatizada de revisão de TCCs, integrada a um Modelo de Linguagem de Grande Escala ou *Large Language Model* (LLM) *open-source*, que permite receber *feedbacks* relacionados a aspectos textuais e estruturais do trabalho, conforme as normas brasileiras vigentes, melhorando a qualidade da escrita, assegurando a coerência lógica e o rigor metodológico, e diminuindo o número de revisões enviadas ao orientador. Para tanto, foi conduzida uma pesquisa sobre o estado atual das abordagens de correção automatizada de trabalhos científicos, especialmente artigos científicos e TCCs. Em seguida, foi proposta a ideia de concepção da ferramenta, com um *workflow* de como o processo de correção deveria operar. A implementação foi realizada na linguagem *Python*, com o auxílio de bibliotecas e *frameworks* de desenvolvimento, apoiada por uma persistência em um banco de dados *MySQL*. Por fim, o sistema foi testado pelos próprios usuários por meio de uma abordagem de avaliação de usabilidade chamada *System Usability Scale* (SUS). A ferramenta demonstrou-se viável quanto ao uso por estudantes de graduação, além de apresentar resultados promissores no uso de LLM *open-source* para atingir revisões precisas e úteis ao usuário.

As principais contribuições deste trabalho são:

1. A criação de uma ferramenta *web* integrada com um LLM *open-source* para auxiliar a correção de TCCs de estudantes de graduação em português, reduzindo o número de revisões necessárias do trabalho;
2. O direcionamento feito ao LLM integrado aos agentes que compõem a ferramenta, utilizando técnicas de engenharia de *prompt* como o *few-shot prompting*.

Este trabalho está estruturado da seguinte maneira: a Seção 2 apresenta os conceitos-chave do embasamento teórico; a Seção 3 trata dos trabalhos correlatos que apoiam a pesquisa; a Seção 4 descreve os procedimentos metodológicos e o método de avaliação; a Seção 5 expõe a concepção da ferramenta e como ela opera; a Seção 6 detalha o perfil dos usuários e discute os resultados obtidos pela avaliação de usabilidade da ferramenta; por fim, a Seção 7 conclui o trabalho e aponta direções para estudos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

A presente seção apresentará a base necessária para o entendimento do sistema proposto neste trabalho. Dentre os assuntos que serão abordados nesta parte do trabalho estão: Trabalho de Conclusão de Curso (TCC), Processamento de Linguagem Natural (PLN), Modelos de Linguagem de Grande Escala e Engenharia de *prompt*.

2.1 Trabalho de conclusão de curso

O TCC integra o grupo de trabalhos acadêmicos presentes na produção científica e, de acordo com a Associação Brasileira de Normas Técnicas (2024), é constituído por normas específicas e princípios gerais para a elaboração desses trabalhos, visando a apresentação, ao final da produção, para a obtenção do grau e do diploma de Bacharel ou Licenciatura. Com isso, dentre os principais tipos de TCCs, o mais comum é utilizar a denominação de monografia para referir-se a um trabalho teórico que se objetiva sobre um determinado assunto (SILVA *et al.*, 2020).

Salomon (2004) defende que a tese principal do curso de graduação é criar uma mentalidade científica necessária para formar o profissional de nível superior e, baseado no artigo de Ramos (2011), um TCC pode ser estruturado em três partes principais:

- **Parte pré-textual:** Identificação e Resumo;
- **Parte textual:** Introdução, Problema, Hipótese, Justificativa, Objetivos, Metodologia, Cronograma/Fluxograma, Recursos necessários, Resultados, Discussões e Considerações finais (conclusão);
- **Parte referencial:** Anexos, Apêndices e Bibliografia.

Por fim, com tais conceitos em mente, Dias e Silva (2009) complementam que elaborar um trabalho desse escopo exige um planejamento cuidadoso, em que o aluno deve se conectar a fundo com o que já foi desenvolvido por outros estudiosos, quais foram seus argumentos e descobrir o que não foi alcançado por eles, além de entender, ao longo do processo, a relação entre a monografia e a pesquisa científica, em que uma é decorrente da outra, não havendo a primeira sem a segunda (SALOMON, 2004).

2.2 Processamento de linguagem natural

O Processamento de Linguagem Natural (PLN), em inglês *Natural Language Processing* (NLP) é um ramo da IA que teve início nos anos 1950 e, hoje em dia, é um domínio complexo e multidisciplinar que busca processar e extrair significado da linguagem humana (NADKARNI *et al.*, 2011). Chowdhary (2020) conceitua o PLN como uma coleção de técnicas computacionais para a análise automática e representação da linguagem humana, motivada pela teoria. Tantos foram os avanços das técnicas adotadas que estas acabaram por resultar na utilização do Aprendizado Profundo (AP), não exigindo que o programador fornecesse as regras explícitas para cada tarefa; o próprio algoritmo deduziria o processo de mapeamento da entrada para a saída (JOHRI *et al.*, 2021).

Qin *et al.* (2024) adicionam que essas mudanças permitiram que tarefas de PLN fossem impulsionadas pela chegada dos grandes modelos de linguagem. Além disso, até mesmo na educação, a aplicação do PLN foi aprimorada, em especial na universidade, por meio da coleta de *feedback* dos alunos, podendo retornar uma resposta customizada baseada nas necessidades de cada um, mostrando que esses LLMs generativos são adaptáveis a inúmeras aplicações no meio acadêmico, permitindo o desenvolvimento de modelos especializados para tarefas específicas

(FUCHS, 2023).

2.3 Modelos de linguagem de grande escala

Os Modelos de Linguagem de Grande Escala, em inglês *Large Language Models* (LLMs), são algoritmos de Aprendizado de Máquina (AM) treinados em grandes conjuntos de dados de texto para gerar conteúdo semelhante ao que os humanos fariam, traduzir idiomas e também responder a várias perguntas. Eles representam o mais recente avanço em modelos de linguagem e no campo de Processamento de Linguagem Natural (PLN) (NAVEED *et al.*, 2023). Atualmente, a arquitetura mais utilizada nos LLMs é a arquitetura *transformer*, e Zhao *et al.* (2023) explica que, devido à sua excelente paralelização e capacidade, é possível escalar os modelos para centenas ou milhares de bilhões de parâmetros, permitindo que eles prevejam com precisão a próxima palavra em uma sentença.

Thirunavukarasu *et al.* (2023) afirmam que toda essa capacidade de compreensão de texto não é atribuída a um treinamento específico para entender a linguagem como os humanos, mas sim a uma abordagem de aprendizado de associações estatísticas entre palavras, desenvolvendo a capacidade de prever qual palavra melhor completa uma frase, revolucionando a área de PLN. Apesar disso, é essencial que haja conhecimento sobre os riscos e desafios do uso, como a geração de textos incorretos e desviados do significado original (alucinações), bem como equívocos em instruções que levem o modelo a respostas ofensivas e tendenciosas (ZHAO *et al.*, 2023).

2.4 Engenharia de prompt

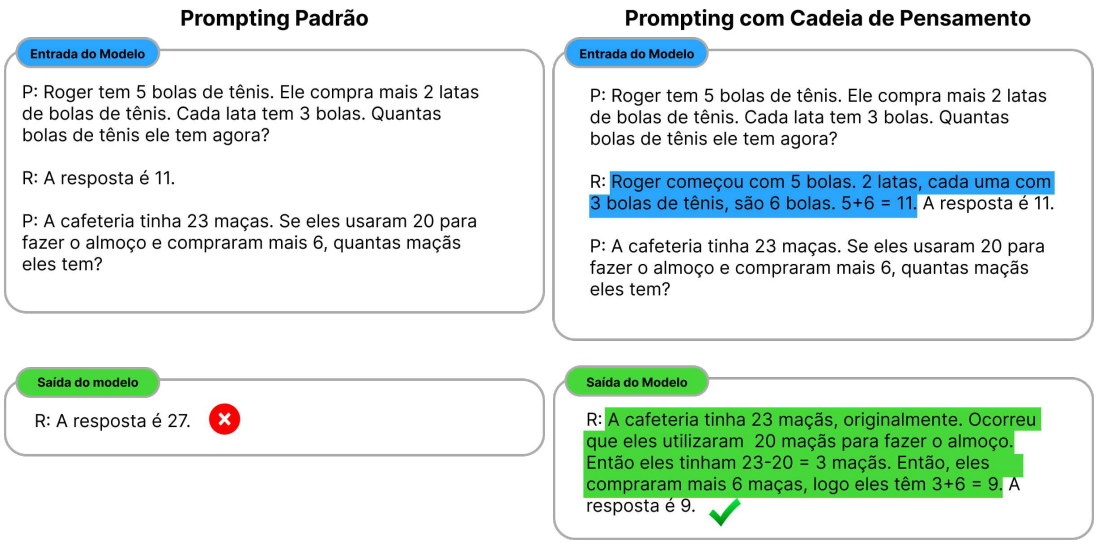
A Engenharia de prompt é uma técnica recente usada pelos modelos generativos de linguagem com o intuito de extrair sua capacidade máxima, utilizando instruções específicas para a geração de respostas precisas. Heston e Khun (2023) veem como um processo crucial para maximizar os benefícios dos modelos, envolvendo o design de entrada (*prompt*) de uma forma que guie o modelo a produzir a saída desejada.

Existem técnicas que aprimoram a resposta do *Large Language Model* (LLM), como o *Few-Shot prompting* [4], que é baseado em um pequeno número de exemplos resolvidos que são fornecidos como parte da entrada para o modelo (REYNOLDS; MCDONELL, 2021). No geral, um *prompt* pode ser organizado como uma instrução completa, fornecendo contexto e até mesmo um indicador de formato de saída. Marvin *et al.* (2023) destaca os principais elementos em um *prompt*:

- **Instrução:** entrada de texto específica que guia o comportamento do modelo;
- **Dados de entrada:** entrada ou pergunta que se deseja que o modelo processe;
- **Contexto:** forma de fornecer contexto de fundo ao modelo para respostas mais relevantes;
- **Indicador de saída:** tipo ou formato da saída desejada (parágrafo curto, história de ficção científica).

[4] https://www.promptingguide.ai/pt/techniques/fewshot

Figura 1 – Exemplo de processo utilizando *Chain-of-thought*.



Fonte: Adaptado de (WEI *et al.*, 2022).

3 TRABALHOS RELACIONADOS

O trabalho de Liang *et al.* (2024) propõe um *pipeline* automatizado baseado no *GPT-4* para a geração de *feedback* sobre artigos científicos completos, organizado em quatro seções: significância e novidade, razões para aceitação, razões para rejeição e sugestões de melhoria. A metodologia envolveu uma análise empírica em larga escala com dois *datasets*: 3.096 artigos aceitos de 15 periódicos da *Nature* e 1.709 artigos da *International Conference on Learning Representations* (ICLR), totalizando 6.505 comentários de revisores humanos entre 2022 e 2023. A avaliação contou com 308 pesquisadores de 110 instituições, das áreas de IA e biologia computacional. Os resultados indicaram sobreposição média de 30,85% (*GPT-4* vs. humano) contra 28,58% (humano vs. humano) nos periódicos da *Nature*, e de 39,23% contra 35,25% na ICLR. Para artigos rejeitados da ICLR, a sobreposição atingiu 47,09%, sugerindo maior utilidade do *feedback* em cenários que demandam revisões substanciais. Os experimentos também evidenciaram que os *feedbacks* não são genéricos, identificando questões recorrentes reconhecidas por múltiplos revisores humanos.

Em Chamoun *et al.* (2024), é apresentada a ferramenta *Scientific WrIting Focused Feedback Tool* (SWIF2T), que utiliza uma arquitetura multiagente composta por *planner*, *investigator*, *reviewer* e *controller*, alavancando instâncias do *GPT-4* para fornecer comentários acionáveis durante a escrita de artigos científicos. A avaliação foi conduzida com 300 unidades extraídas dos *datasets NLPeer*, *F1000rd* e *ARIES*, contendo revisões humanas completas e citações de fraquezas em trechos específicos. A principal métrica retrospectiva, a taxa de sobreposição, mostrou valores de 30,85% para periódicos da *Nature* e 39,23% para a ICLR, comparáveis aos observados entre revisores humanos. Em um estudo prospectivo, 57,4% dos 308 pesquisadores consideraram o *feedback* útil ou muito útil, e 82,4% o avaliaram como mais benéfico do que o de ao menos alguns revisores humanos. Apesar de tendências recorrentes nas sugestões, os resultados indicam que o *feedback* gerado por LLMs pode complementar a avaliação humana.

O trabalho de D'Arcy *et al.* (2024) propõe o *Multi-Agent Review Generator* (MARG), uma ferramenta multiagente baseada em múltiplas instâncias do *GPT-4*, composta por agentes *leader*, *workers* e *experts*, além de uma variante aprimorada, o *Multi-Agent Review Generator with Specialized Agents* (MARG-S). Avaliado com 30 artigos do corpus *ARIES*, o MARG-S demonstrou maior robustez em relação ao MARG, reduzindo a taxa de comentários genéricos de 60% para 29%. Foram gerados, em média, 3,7 comentários bons por artigo, com 71% classificados como precisos e específicos. Na avaliação automatizada, o MARG-S superou todas as *baselines* em *recall*, com valor de 15,84, além de apresentar a maior quantidade de comentários gerados (19,8 em média), reforçando a eficácia de arquiteturas multiagente em tarefas complexas de revisão por pares.

Em Idahl e Ahmadi (2025), é apresentado o *OpenReviewer*, um sistema *open-source* baseado no modelo especializado *Llama-OpenReviewer-8B*, ajustado com 79.000 revisões de alta qualidade provenientes da ICLR e da *Conference on Neural Information Processing Systems* (NeurIPS). O modelo foi treinado a partir de 36.000 artigos e 141.000 revisões, utilizando uma janela de contexto de até 128.000 *tokens* e um processo de *fine-tuning* com duração aproximada de 34 horas. A avaliação considerou 400 artigos de teste e indicou que o *OpenReviewer* apresentou maior alinhamento com revisões humanas em 55,5% dos casos. No método *LLM-as-a-judge*, utilizando o *GPT-4o* como avaliador, o sistema obteve taxas de vitória entre 60% e 76% em comparação com outros LLMs, evidenciando que modelos *open-source* especializados podem produzir revisões acadêmicas críticas e realistas, mesmo com um número reduzido de parâmetros.

Após uma apresentação de exemplos de trabalhos que exploram a temática de LLMs

aplicados a artigos científicos para a geração de *feedback*, agora é possível discernir algumas nuances e distintas contribuições que cada um desses estudos oferece. O presente trabalho inova ao utilizar um modelo *open-source* focado em TCCs em português, optando pelo uso de um LLM *open-source* gratuito, totalmente voltado para o uso de alunos de graduação. Além disso, a ferramenta desenvolvida neste trabalho será avaliada por sua usabilidade, utilizando a métrica de usabilidade *System Usability Scale* (SUS). A Tabela 1 apresenta a comparação entre os estudos.

Pontos analisados:

1. *Feedback* automatizado em trabalhos científicos;
2. Uso de LLM *open-source*;
3. Uso de *prompts especializados*;
4. Arquitetura multiagente;
5. Geração de *feedback* em português brasileiro;

Tabela 1 – Tabela comparativa entre as contribuições deste trabalho em relação aos outros apresentados.

Autores	**Solução utilizada**	**Métricas**	**1**	**2**	**3**	**4**	**5**
Liang *et al.* (2024)	*Pipeline* automatizada que gera *feedback* estruturado sobre artigos científicos utilizando *GPT-4*.	Taxa de Sobreposição Simkiewicz-Simpson, Índice Jaccard e Coeficiente Sørensen–Dice.	Sim	Não	Sim	Não	Não
Chamoun *et al.* (2024)	Sistema que gera feedback focado, específico e acionável, identificando fraquezas em artigos científicos.	Similaridade Textual (*METEOR, BLEU@4, ROUGE-L*), Precisão do rótulo de aspecto e *F1-Score*.	Sim	Não	Sim	Sim	Não
D'Arcy *et al.* (2024)	Múltiplos agentes de LLM para gerar *feedback* de revisão por pares específico. Útil para artigos científicos longos.	*Recall*, *Precision*, Jaccard, Comentários considerados "Bons"por Artigo e Especificidade de Comentários.	Sim	Não	Sim	Sim	Não
Idahl e Ahmadi (2025)	Abordagem com LLM open-source que gera revisões críticas e estruturadas de artigos de ML/IA.	Correspondência exata da recomendação, Erro absoluto médio, Pontuação Média (Criticidade) e *Win Rate* em *Review Arena*.	Sim	Sim	Sim	Não	Não
Este trabalho	Arquitetura multiagente com LLM *open-source* e *prompts* especializados que geram *feedbacks* de TCCs em português.	Métrica de usabilidade com o SUS e uma *baseline* comparativa entre o *feedback* gerado de LLMs com *zero-shot* e a abordagem *open-source* aprimorada.	Sim	Sim	Sim	Sim	Sim

Fonte: elaborada pelo autor.

4 METODOLOGIA

Este capítulo trata da metodologia que foi utilizada para a construção e avaliação da ferramenta de revisão automatizada de TCCs em português. A Seção 4.1 apresenta as tecnologias que a ferramenta possui; a Seção 4.2 explica o conjunto de dados utilizado para a validação das revisões; a Seção 4.3 justifica os motivos da escolha do modelo *open-source* e a Seção 4.4 trata da forma como a ferramenta foi avaliada.

4.1 Tecnologias e implementação da aplicação

Inicialmente, a ferramenta chamada de *ARAMIS* [5] teve suas telas prototipadas no *Figma*[6]. A implementação focou na linguagem de programação *Python*, aproveitando-se de *frameworks* de desenvolvimento *web*, manipulação de dados, requisições e também na construção dos agentes. As tecnologias utilizadas se dividiram em um *front-end* construído com o *Streamlit* [7], um *framework* que permite criar uma interface gráfica amigável e intuitiva, facilmente integrável ao *back-end*, aproveitando-se das chamadas *Hypertext Transfer Protocol* (HTTP), que foram elaboradas com a tecnologia *FastAPI*, auxiliadas pelo *framework Agno*[8] para a construção dos agentes, que possui suporte à *Application Programming Interface* (API) do *HuggingFace* [9], conectando-se facilmente ao LLM *open-source*.

O sistema utiliza um banco de dados *MySQL*, em virtude da simplicidade de configuração, realização de testes e armazenamento dos dados em um banco relacional. O *MySQL* armazena os dados dos usuários cadastrados na plataforma e as 10 últimas revisões de cada usuário — permitindo consulta futura. Ciente de suas limitações, a ferramenta armazena apenas dados mais simples, como as informações dos usuários e as revisões geradas pelo modelo, não tendo previsão de alocar além do escopo definido neste trabalho.

Todas as tecnologias citadas anteriormente foram implementadas em suas versões mais recentes, até o momento em que foi possível manter a compatibilidade e o funcionamento delas. Portanto, adotou-se o *Python 3.11*. No *front-end*, foi utilizado o *Streamlit 1.52.2*; no *back-end*, o *FastAPI 0.124.4*; O *framework Agno* foi implementado na versão 2.3.13 e, para o gerenciamento do banco de dados, o *phpMyAdmin 5.2.3*, definindo, assim, a construção do *Minimum Viable Product* (MVP) do ARAMIS.

4.2 Conjunto de avaliação

Para aprimorar a qualidade da revisão pelos agentes que compõem o ARAMIS, um conjunto para avaliação foi selecionado, com 8 TCCs aprovados pelos alunos de graduação da Universidade Federal do Ceará (UFC) do campus de Crateús, especificamente da área de computação, dos cursos de Ciência da Computação (CC) e Sistemas de Informação (SI), entre o período de 2024 e 2025. Esses trabalhos foram coletados a partir do Repositório Institucional[10] da universidade, devidamente autorizados por seus autores para uso acadêmico.

O conjunto de TCCs foi extraído em formato *Portable Document Format* (PDF) e passou por um pré-processamento antes de ser utilizado pelo modelo. O conjunto passou por

[5] Disponível em http://enginelab.ufc.br/
[6] Disponível em https://figma.com/
[7] Documentação do Streamlit - https://docs.streamlit.io/
[8] Documentação do Agno - https://docs.agno.com/
[9] HuggingFace - https://huggingface.co/
[10] Disponível em https://repositorio.ufc.br/

um *script Python*, convertendo-o por meio da biblioteca *Marker* para o formato *Markdown*, um formato que mantêm uma estrutura com marcadores, preservando a estrutura hierárquica do documento. A qualidade da conversão foi avaliada manualmente e, da mesma forma, corrigida (quando necessário) pela separação das seções e pela qualidade das tabelas, de acordo com a organização padrão dos TCCs. A tentativa foi aderir a uma abordagem mais precisa e, além disso, que pudesse lidar bem com tabelas e fórmulas matemáticas, o que é o caso do *Marker*, já que estas são altamente presentes em TCCs da computação.

A escolha por um *corpus* de TCCs da área supracitada justifica-se pela afinidade com o objetivo deste trabalho e pela delimitação de escopo definida no planejamento, visando a maior precisão nas revisões. Uma *pipeline* foi definida com a extração e conversão dos TCCs, comportando o texto por completo, preservando a estrutura de capítulos e seções, destacando a marcação dessas partes do texto, com o intuito de conferir um peso maior ao rigor de sua análise pelo modelo, mas ignorando ou removendo algumas informações não pertinentes, como a identificação dos autores, orientadores, dedicatórias, agradecimentos, entre outros. Essa remoção foi efetuada após os arquivos concluírem a conversão para o formato com extensão "*.md*"e, após isso, uma revisão manual foi feita, identificando algumas inconsistências, como diversas referências listadas como "*(#page-54-4)*", capítulos sendo marcados com uma sequência de três ou quatro cerquilhas, em vez de apenas uma, visto que em *Markdown*, uma quantidade menor de "*#*"indica maior prioridade sobre o texto, e até mesmo figuras que, ao serem convertidas, não faziam nenhum sentido, virando lixo e tendo que ser removidas.

Figura 2 – Representação do *pipeline* de extração e limpeza dos TCCs

Fonte: elaborada pelo autor.

Foi efetuado um cálculo a quantidade de *tokens* que um documento de TCC ocupa por completo; por haver diferentes tamanhos, a Tabela 2 exibe a quantidade de *tokens* que cada um possuía antes e depois do tratamento. A quantidade de *tokens* foi contada a partir da plataforma *Tokenizer*[11], pertencente a *OpenAI*; logo, nota-se que a noção de quantidade é baseada na contagem realizada nos modelos *Generative Pre-Trained Transformers* (GPT), especificamente nas versões 4o e 4o mini. Foram testadas outras plataformas como *Token Calculator*[12], e a diferença não ultrapassou 5.000 *tokens* em comparação com o processo de tokenização utilizado por outros LLMs.

[11] Disponível em https://platform.openai.com/tokenizer
[12] Disponível em https://token-calculator.net/

Tabela 2 – Quantidade de tokens antes e depois do tratamento nos trabalhos extraídos

Trabalhos	Qtd. tokens antes	Qtd. tokens depois	Redução (%)
Trabalho 1	25.959	20.157	22,35%
Trabalho 2	31.275	21.341	31,76%
Trabalho 3	30.536	20.812	31,84%
Trabalho 4	11.210	6.727	39,99%
Trabalho 5	19.460	11.811	39,31%
Trabalho 6	29.713	18.762	36,86%
Trabalho 7	38.050	22.314	41,36%
Trabalho 8	20.678	11.939	42,26%

Fonte: elaborada pelo autor.

Após o tratamento, obteve-se uma redução média de 35,71% na quantidade de *tokens*. Feito isso, seções específicas dos TCCs foram utilizadas para validar a qualidade da revisão gerada pelos agentes integrados ao LLM *open-source*, explicados na Seção 4.3.

4.3 Escolha do LLM open-source

A escolha do modelo baseou-se em testes preliminares, comparando modelos proprietários e de código aberto (*open-source*). Foram testados: *Gemini 2.5 Pro*, *GPT-4o*, *GPT-4o-mini* e os *open-source gpt-oss-20b*, *meta-llama-3.1-8b-instruct* e *deepseek-r1-distill-qwen-14b*. Realizaram-se testes para validar a saída dos modelos, seguindo um critério de inserção do texto integral do TCC em formato *Markdown*, com o modelo identificando, por si só, a parte desejada do texto (ou seja, o arquivo .md do conjunto de validação foi inserido e o próprio modelo deveria identificar a seção definida na pré-configuração). Considerou-se se o modelo alucina ao analisar o conjunto de validação com os TCC pré-processados, o tempo em que a revisão foi gerada e outros critérios. A opção de comprimento de contexto foi respeitada para estar sempre com valor máximo, evitando possíveis limitações na janela de contexto. A temperatura (parâmetro responsável por controlar a aleatoriedade dos textos gerados pelo modelo) foi configurada sempre com o valor de 0,8 (esse valor costuma variar de 0 até 2, a depender do modelo) quando possível, para que as revisões sejam coerentes, mas criativas, de certa forma.

O *script* inicial dos agentes — com o modelo salvando as revisões em arquivo — rodou em um computador com um processador Intel i7-12650h, 32GB de RAM e um SSD NVMe de 1TB, com Ubuntu 24.04. Os LLMs proprietários foram conectados a partir de sua API oficial, e os *open-source*, por meio de uma conexão local com o aplicativo *LM Studio* instalado em outra máquina, esta com um processador AMD Ryzen 7 7700, 64GB de RAM, um SSD NVMe de 2TB, com Ubuntu 24.04 e duas placas de vídeo *NVIDIA* RTX 5090, de 32GB de VRAM, cada.

Após a obtenção dos resultados anteriormente citados, o modelo que apresentou melhor desempenho foi o *gpt-oss-20b*[13] da OpenAI, integrado à ferramenta por meio da API do *HuggingFace*. A Tabela 3 apresenta os dados obtidos nos testes com modelos proprietários e *open-source*. Foram realizados 10 testes com cada TCC do conjunto de validação, permitindo calcular o tempo médio para a geração da revisão, a quantidade média de comentários produzidos e a taxa média de alucinações observadas. Os testes consistiram em aplicar integralmente os

[13] https://lmstudio.ai/models/openai/gpt-oss-20b

TCCs aos modelos, de modo que estes identificassem a seção solicitada no *prompt* e retornassem a revisão correspondente.

Tabela 3 – Média obtida pelos modelos proprietários e *open-source* testados

Modelo	**Tempo**	**Qtd. comentários**	**Taxa de alucinação**
Gemini 2.5 Pro	102.22 ± 45.11 s	8	0
GPT-4o	70.30 ± 22.46 s	7	4
GPT-4o-mini	56.30 ± 25.89 s	7	6
gpt-oss-20b	**54.30 ± 23.09 s**	**10**	**2**
meta-llama-3.1-8b-instruct	28.00 ± 12.80 s	5	8
deepseek-r1-distill-qwen-14b	39.30 ± 15.44 s	6	5

Fonte: elaborada pelo autor.

4.4 Avaliação da ferramenta

A avaliação deste trabalho foi feita sobre a qualidade da usabilidade da ferramenta produzida. O método foi o *System Usability Scale* (SUS), desenvolvido por Brooke (1995), uma métrica escolhida por ser amplamente reconhecida para medir avaliações subjetivas de usabilidade em uma ampla gama de contextos, inclusive na avaliação de sistemas de informação. Além disso, conforme Padrini-Andrade *et al.* (2019), essa possibilidade de avaliar diversos tipos de contextos deve-se ao fato de que o SUS é tecnologicamente agnóstico, ou seja, possui uma abordagem neutra e aberta. Os usuários avaliaram a usabilidade do ARAMIS imediatamente após a conclusão das tarefas no sistema, para que não houvesse discussões ou reflexões que pudessem enviesar a opinião. O questionário SUS consiste em 10 perguntas, funcionando de acordo com a Figura 3, onde os usuários responderam em uma escala *Likert* que varia de 1 a 5 (JOSHI *et al.*, 2015), em que 1 representa *"Discordo Completamente"* e 5 representa *"Concordo Completamente"*. A Figura 3 ilustra as perguntas que o questionário contém no processo de avaliação e exibe a escala de avaliação.

Figura 3 – Exemplo do questionário SUS que será utilizado

Questionário System Usability Scale	**Discordo Completamente**				**Concordo Completamente**
1. Acho que gostaria de usar este sistema com frequência.	1	2	3	4	5
2. Achei o produto desnecessariamente complexo	1	2	3	4	5
3. Achei o sistema fácil de usar	1	2	3	4	5
4. Acho que precisaria do suporte de um técnico para poder usar este sistema	1	2	3	4	5
5. Achei que as várias funções deste sistema estavam bem integradas	1	2	3	4	5
6. Achei que havia muita inconsistência neste sistema	1	2	3	4	5
7. Imagino que a maioria das pessoas aprenderia a usar este sistema muito rapidamente	1	2	3	4	5
8. Achei o sistema muito complicado de usar	1	2	3	4	5
9. Eu me senti muito confiante ao usar o sistema	1	2	3	4	5
10. Precisei aprender muitas coisas antes de começar a usar esse sistema.	1	2	3	4	5

Adaptado de (BROOKE, 1995).

As perguntas presentes no artigo original estavam em inglês e foram traduzidas para o português. Após isso, foram inseridas em um formulário na plataforma *Google Forms*, com uma seção para identificação, uma para responder às perguntas do SUS e uma de sugestões sobre a ferramenta, que foram preenchidas pelos usuários que participaram da pesquisa de avaliação de usabilidade.

4.4.1 Cálculo do SUS

Após a conclusão do questionário, o SUS avalia a métrica por meio de um cálculo para consolidar o resultado, que varia de 0 a 100. O método utilizado para a pontuação resultou em um número único que compõe a usabilidade geral do sistema avaliado. Para consolidar esse cálculo, ele seguiu a abordagem de seu artigo de origem em Brooke (1995), que envolve a seguinte metodologia: os itens ímpares (Alternativas Positivas) tem sua nota de atribuição subtraída por 1, enquanto para os itens pares (Alternativas Negativas), o cálculo é feito subtraindo-se o valor 5 da nota de atribuição. Após a conclusão, todas as pontuações das dez perguntas são somadas e o resultado é multiplicado por 2,5. O processo de cálculo do SUS é exemplificado na Fórmula 1 a seguir.

$$SUS = 2{,}5 \times \left\{ \left[\sum_{i=impar} (x_i - 1) \right] + \left[\sum_{i=par} (5 - x_i) \right] \right\} \tag{1}$$

em que:

- Para itens **ímpares** (1, 3, 5, 7, 9): $x_i - 1$
- Para itens **pares** (2, 4, 6, 8, 10): $5 - x_i$
- x_i é a nota do usuário para cada item (escala de 1 a 5)

Brooke (1995), em seu artigo, não definiu uma escala concreta de pontuação para determinar se a usabilidade seria boa ou ruim. Por isso, este trabalho seguiu a interpretação defendida por Bangor *et al.* (2009), que resolve essa lacuna deixada pelo trabalho de John Brooke (BROOKE, 1995). A abordagem apresentada é a seguinte:

- **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 abaixo de 60.

5 PROPOSTA

Neste capítulo, é apresentado o sistema de auxílio à correção de TCCs desenvolvido neste trabalho, com o objetivo de assegurar a coerência lógica e o rigor metodológico dos textos, diminuindo o número de revisões enviadas ao orientador.

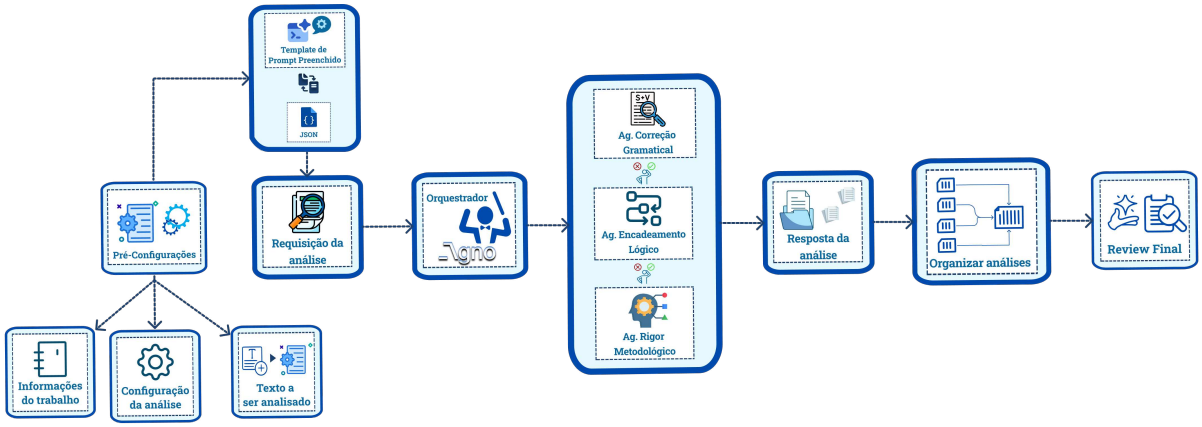
5.1 ARAMIS

O sistema concebido neste trabalho é chamado de *Academic Review Agents for Methodological Improvements* (ARAMIS), uma plataforma direcionada a alunos de graduação em processo de desenvolvimento do Trabalho de Conclusão de Curso (TCC) em português. É composta por agentes específicos, permitindo a interação entre o usuário e uma interface de inserção de dados. Esta interface recebe algumas informações do texto do TCC do aluno, a qual retorna uma revisão sobre a seção do trabalho inserida. Para que funcione, a aplicação é composta por agentes integrados a um LLM *open-source* voltados exclusivamente para tratar de assuntos ligados ao TCC em questão. O ARAMIS tem agentes que atuam em níveis que se complementam — correção gramatical, encadeamento lógico e rigor metodológico, com atribuições internas entre esses três agentes —, ativando-se a partir da configuração do usuário, com a intenção de fornecer *feedbacks* precisos e úteis sobre o texto do trabalho em desenvolvimento.

5.1.1 Arquitetura e fluxo do ARAMIS

A seguir, é proposto o modelo da arquitetura do ARAMIS, separado em cinco etapas principais bem definidas, ilustradas na Figura 4, desde a pré-configuração até a organização final das revisões feitas pelo LLM.

Figura 4 – A figura apresenta o fluxograma do ARAMIS, detalhando suas principais etapas.



Fonte: elaborada pelo autor.

Para acessar o ARAMIS, o usuário deve ter um cadastro na plataforma. Após o login, é apresentada a *dashboard*, exibida na Figura 5, onde há quatro opções de acesso: a página inicial; o acesso principal à funcionalidade de correção de TCCs; a seção de histórico das revisões; a seção de gerenciamento do perfil; e a seção de informações úteis sobre a ferramenta.

Figura 5 – A figura apresenta a tela inicial do ARAMIS e as opções disponíveis na *sidebar*.

Fonte: elaborada pelo autor.

Na seção Nova Correção, como mostra a Figura 6, é exibido um formulário que o usuário preenche com as informações de seu trabalho, fazendo parte da interface com as opções de pré-configuração, ilustradas no fluxograma da Figura 4, auxiliando na resposta que o LLM deve retornar e direcionando-o a fornecer uma resposta precisa e estruturada. São as opções de pré-configuração:

- Inserir o título e a área de conhecimento do TCC;
- Selecionar a seção a que se refere o texto inserido pelo usuário;
- Definir entre um e três agentes que podem ser acionados para gerar o *feedback*;
- Definir o nível de rigor a qual o modelo deverá operar

Na terceira etapa, baseada nas informações preenchidas na pré-configuração, o serviço de análise se responsabiliza por receber os valores inseridos no *front-end*, processá-los e preencher os *templates* de *prompts*, convertendo-os para o formato *JavaScript Object Notation* (JSON) e, assim, enviá-los ao orquestrador *Agno*, responsável pela comunicação entre o *back-end* e a chamada ao serviço do LLM hospedado no *Hugging Face*, por meio de sua *API*, iniciando, assim, o processo de revisão do texto do TCC.

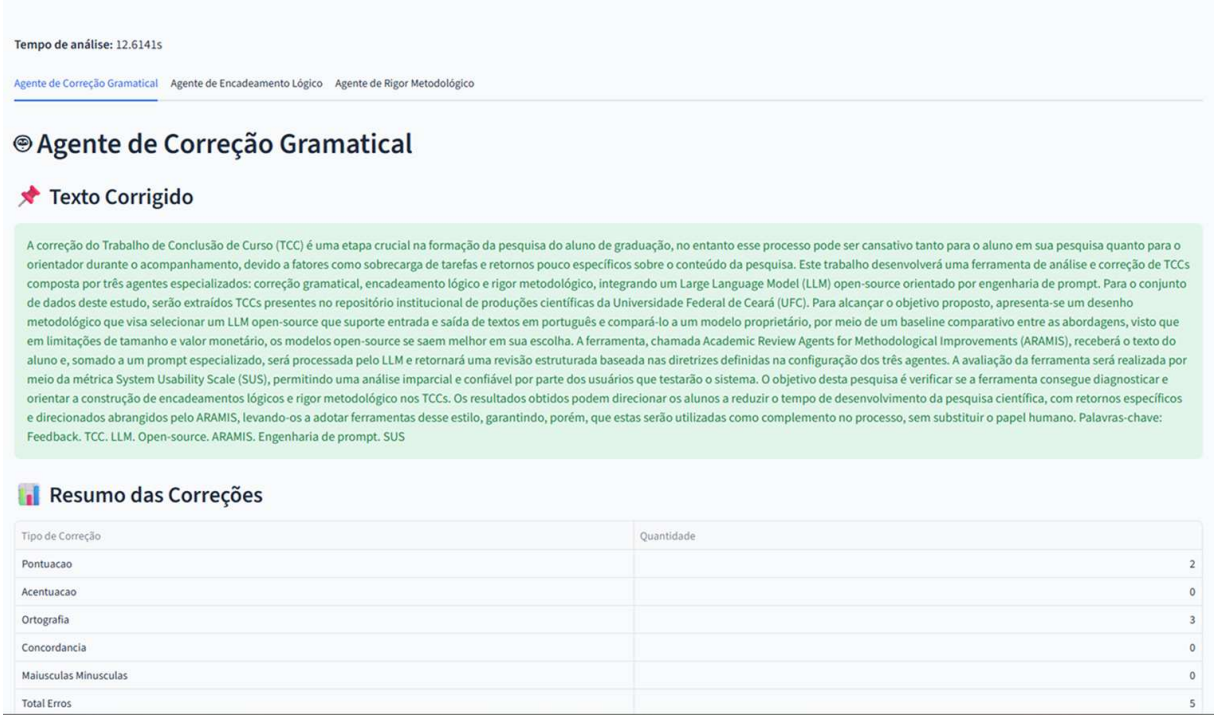
Figura 6 – A figura apresenta o formulário de preenchimento com as informações de pré-configuração do ARAMIS.

Fonte: elaborada pelo autor.

O *framework Agno* tem como função coordenar como os agentes são definidos e como agem. Dessa forma, a definição dos três agentes é semelhante, iniciando sua operação quando o processamento de recebimento de dados do *front-end* é concluído, recebendo a escolha do usuário na pré-configuração e ativando-os conforme selecionados, entre outros. A atividade operacional de cada agente é sequencial, isto é, caso os três agentes sejam acionados, o próximo só começará a operar após a conclusão da operação do agente anterior, devido a existência de apenas um LLM integrado à ferramenta, funcionando para apenas uma consulta por vez. Caso apenas um agente seja acionado, os demais não deverão operar, permanecendo inativos durante toda a rodada do fluxo do ARAMIS.

Por fim, o resultado das revisões é sequenciado em uma única saída JSON (caso apenas um agente seja acionado, há apenas uma revisão, também em uma saída), separado pela resposta de cada agente e, após isso, as revisões são exibidas na interface do usuário, ilustrado na Figura 7, onde cada uma é separada em sua própria seção, permitindo a visualização detalhada da correção feita por cada modelo integrado aos agentes.

Figura 7 – A figura apresenta uma revisão gerada pelo ARAMIS, na seção de revisões do usuário.



Fonte: elaborada pelo autor.

5.2 Função dos agentes do ARAMIS

O ARAMIS, como uma ferramenta multiagente, é composto por um módulo de agentes, que são funções construídas em *Python*, auxiliadas pelo *framework* de orquestração multiagente *Agno*, utilizando-o para facilitar a configuração e o comportamento dos agentes que seriam utilizados em uma rodada de execução do programa, ou seja, de acordo com o fluxo do ARAMIS, exibido na Figura 4. Dessa forma, dependendo da pré-configuração do usuário, o *Agno* decide qual agente é considerado para a função designada, quando necessário, recebendo uma chamada composta por um corpo de informações passado na requisição.

Por meio do *framework Agno*, os três agentes são controlados e cada um está integrado em instâncias semelhantes do mesmo LLM *open-source*, o *gpt-oss-20b*, da *OpenAI*. A Tabela 4 descreve as características que cada agente possui e o tipo de particularidades textuais que cada um têm enfoque.

5.3 Modelagem dos prompts

Os *prompts* foram estruturados de maneira a receber orientações explícitas sobre quem são, o papel a desempenhar, os detalhes e o tipo de resposta a retornar. Além disso, há variáveis (*placeholders*) que o sistema preenche automaticamente com informações inseridas pelo usuário na etapa anterior à ativação da revisão. Essas variáveis contêm, por exemplo, o texto da seção submetida, a área específica do trabalho e o tipo de avaliação desejada. Essas variáveis estão dentro de chaves, onde serão automaticamente substituídas no momento da execução da LLM, conforme as escolhas do usuário (esclarecido na Subseção 5.1.1). Existirão variáveis padronizadas em todos os *prompts*, pois todos seguirão uma estrutura semelhante, com a diferença do direcionamento imputado a cada um deles. Abaixo, detalhamos quais são essas variáveis que estarão entre chaves para serem substituídas no momento da entrada do modelo:

Tabela 4 – Resumo das funções dos agentes do sistema ARAMIS

Agente	Descrição
Correção gramatical	Agente que utiliza técnicas de Engenharia de *prompt*, como o *few-shot* explicado na Seção 2.4, voltado especificamente para identificar e listar erros ortográficos e possíveis equívocos de acentuação que o texto inserido pode apresentar. Está orientado a não alterar o estilo de escrita do autor, apenas expor a quantidade de erros e suas correções; sua saída foi configurada para também exibir o texto corrigido.
Encadeamento lógico	Agente preparado para lidar com a coerência entre sentenças e seções presentes no trabalho. O modelo realiza uma análise em nível micro (sentenças adjacentes/parágrafos da seção). Também é utilizado um *prompt* direcionado, com a técnica de *few-shot*, estruturando e aprimorando a resposta gerada pelo modelo.
Rigor metodológico	Agente construído com maior atenção, contendo um *prompt* próprio direcionado que utiliza técnicas estruturadas por etapas e é testado com abordagens como o *few-shot*. Sua atuação concentra-se em verificar a coerência entre o problema de pesquisa, os objetivos, o método e a análise de resultados, em especial no capítulo de Metodologia.

Fonte: elaborada pelo autor.

- *secao_desejada*: Indica nos *prompts* a seção que será abrangida para análise. Foi definido ser apenas uma, após notar-se uma melhoria das revisões ao abarcar apenas uma seção por vez. Essa variável no *prompt* ajuda o modelo a ser mais específico sobre a seção selecionada;
- *titulo_tcc*: Contém o título do trabalho do aluno. Desta forma, o LLM sabe especificamente o tema que está tratando e as informações possivelmente atreladas a ele, adquirindo um contexto mais preciso;
- *area_conhecimento_tcc*: Representa a área do conhecimento a qual o TCC está atrelado. Essa variável ajuda o LLM a entender por que este assunto está sendo escrito com aquele conteúdo e daquela maneira;
- *nivel_rigor_modelo*: Define o nível de rigor que o modelo deve avaliar aquele trecho do TCC do aluno. Essa variável serve para apoiar o teor da revisão do LLM de acordo com o desejo do usuário;

Esse design específico de *prompt* teve a intenção de explorar a capacidade do LLM de capturar padrões complexos, como a objetividade e o rigor científico. Esse enfoque é especialmente útil para os modelos de linguagem, que possuem um forte potencial de generalização e podem identificar padrões globais em uma única leitura dos textos.

6 RESULTADOS

Este capítulo caracteriza o perfil das pessoas que participaram dos experimentos e expõe os resultados dos usuários reais da ferramenta ARAMIS, por meio do método de avaliação de usabilidade SUS, além dos *feedbacks* positivos e negativos deixados pelas pessoas que avaliaram o sistema.

6.1 Perfil dos usuários

A avaliação da ferramenta foi realizada por 10 graduandos da Universidade Federal do Ceará (UFC), dos cursos de Ciência da Computação (CC) e Sistemas de Informação (SI), todos em processo de escrita do TCC 1 ou TCC 2. Esses graduandos tiveram a ferramenta à disposição na *web* para que inserissem partes de seus TCCs em um horário oportuno e, dessa forma, obtiveram o resultado da análise dos agentes do ARAMIS. Após os testes, os participantes responderam a um formulário na plataforma *Google Forms*, que continha as perguntas de usabilidade do SUS. Os usuários foram orientados a preencher com o nome, curso e semestre e, de acordo com a ordem de resposta do formulário, os resultados foram organizados em uma tabela. Para o relato neste trabalho, os dados sensíveis foram anonimizados, ficando, por exemplo: Usuário CC 1, Usuário SI 2, etc. A Tabela 5 demonstra o perfil dos participantes da pesquisa de usabilidade do ARAMIS.

Tabela 5 – Perfil dos usuários que avaliaram o ARAMIS

Participante	Gênero	Escolaridade	Universidade - Campus	Semestre
Usuário SI 1	Masculino	Ens. Médio Completo	UFC - Crateús	12º
Usuário CC 1	Masculino	Ens. Médio Completo	UFC - Crateús	10º
Usuário SI 2	Masculino	Ens. Médio Completo	UFC - Crateús	10º
Usuário CC 2	Masculino	Ens. Médio Completo	UFC - Crateús	12º
Usuário CC 3	Masculino	Ens. Médio Completo	UFC - Crateús	10º
Usuário SI 3	Feminino	Ens. Médio Completo	UFC - Crateús	8º
Usuário CC 4	Masculino	Ens. Médio Completo	UFC - Crateús	10º
Usuário CC 5	Masculino	Ens. Médio Completo	UFC - Crateús	8º
Usuário SI 4	Masculino	Ens. Médio Completo	UFC - Crateús	10º
Usuário CC 6	Feminino	Ens. Médio Completo	UFC - Crateús	8º

Fonte: elaborada pelo autor.

6.2 Resultado dos usuários

A Tabela 6 exibe a nota média das perguntas *pares*[14] e ímpares, e também os resultados finais da avaliação SUS de acordo com o cálculo explicado na Subseção 4.4.1.

[14] Há uma diferença na exibição da média das perguntas pares: quanto mais próximo de 1, melhor avaliada foi a pergunta, pois são alternativas com teor negativo e, dessa forma, quanto mais próximo de 5, pior seria. Já nas ímpares, aplica-se a escala *Likert* normalmente, em virtude das perguntas possuírem teor positivo, e quanto mais próximo de 5, melhor seria.

Tabela 6 – Resultados obtidos pelos usuários que avaliaram o ARAMIS

Usuários	Média perguntas pares	Média perguntas ímpares	Nota final do usuário
Usuário SI 1	1	4,2	90
Usuário CC 1	1,6	4,2	80
Usuário SI 2	1,2	4,8	95
Usuário CC 2	1,8	5	90
Usuário CC 3	1,2	4,6	92,5
Usuário SI 3	1,8	5	90
Usuário CC 4	1	5	100
Usuário CC 5	1,2	4	85
Usuário CC 6	1,4	4,2	85
Usuário SI 4	1,2	4,8	95
Nota média final da usabilidade: 90,5			

Fonte: elaborada pelo autor.

Os resultados apresentados mostram que o ARAMIS, na avaliação dos usuários, obteve a nota média de 90,5 pontos e, de acordo com a interpretação de Bangor *et al.* (2009), a ferramenta foi classificada como excelente. Nenhuma nota final esteve abaixo de 80 pontos, implicando que nenhum usuário considerou sua usabilidade abaixo da escala considerada boa. Além disso, as médias de todas as perguntas foram elevadas, o que indica que quase todos os usuários consideram que o ARAMIS possui uma usabilidade acima da média, em virtude do que é exibido na lista de perguntas da Figura 3, sendo uma ferramenta útil para todos os que a avaliaram.

Os usuários, em seu *feedback*, indicaram a simplicidade de uso da ferramenta, a boa construção das seções, a qualidade e a precisão das revisões dos agentes, em especial 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 ter um *feedback* sobre como está o desenvolvimento do trabalho e no que ele precisa melhorar. Por fim, aqueles que estavam tanto no TCC 1 quanto no TCC 2 relataram a intenção de continuar utilizando o ARAMIS para revisar seus trabalhos, não se limitando apenas aos testes anteriormente realizados.

Embora os resultados tenham sido positivos, ressaltam-se algumas inconsistências identificadas pelos usuários, como problemas pontuais no fluxo de autenticação que exigiram intervenção administrativa, indicando que a estabilidade do sistema precisa de refinamento para escala comercial. O modelo apresentou episódios de alucinação leve, sinalizando erros em palavras inexistentes no texto original ou apontando falhas metodológicas cujas justificativas estavam presentes em outras seções.

6.3 Limitações

Apesar dos resultados promissores, as principais limitações deste trabalho estão relacionadas ao número reduzido de modelos *open-source* analisados, pois apenas seis modelos, majoritariamente de código aberto, representam uma quantidade baixa de exemplos. O número restrito de usuários que participaram da avaliação, conduzida com apenas dez participantes de

uma única instituição, pode limitar a generalização dos resultados. Além disso, as limitações da licença gratuita da API do *HuggingFace* se esgotaram rapidamente durante as chamadas, e o uso de APIs externas gratuitas impôs restrições de latência e volume de requisições. Soma-se a isso a limitação da infraestrutura computacional do servidor onde a ferramenta está hospedada, que não suporta modelos de grande porte, como o *gpt-oss-20b*, o que impactou o desempenho e obrigou a adoção de serviços externos, mesmo tratando-se de modelos gratuitos. Diante desses fatores, é evidente a necessidade de estudos futuros com maior diversidade de modelos, ampliação da base de usuários, maior disponibilidade de recursos computacionais, além do *fine-tuning* de modelos voltados às normas da ABNT e da adoção de inferência local, visando reduzir a dependência de serviços externos e preservar a privacidade dos dados.

7 CONCLUSÕES E TRABALHOS FUTUROS

A correção automatizada de TCCs com LLM *open-source* é importante para otimizar o fluxo de trabalho do aluno durante a produção de seu trabalho, mas também para aprimorar a proficiência do aluno durante o processo de pesquisa. Nesse contexto, uma ferramenta *web* que possibilita a correção de etapas específicas do TCC em língua portuguesa é apresentada como uma oportunidade para diversas soluções de problemas referentes à escrita científica.

A presente pesquisa demonstrou que o emprego de LLMs *open-source*, orquestrados por uma arquitetura multiagente, em que os agentes foram direcionados a funções diferentes, constitui uma solução viável e robusta para a revisão automatizada de TCCs em língua portuguesa, inclusive ao ser integrado à uma ferramenta *web* para uso institucional. Dessa forma, a questão de pesquisa proposta, que investigou *como sistemas de Inteligência Artificial especializados em textos acadêmicos em língua portuguesa podem diagnosticar e orientar a construção de encadeamentos lógicos e o rigor metodológico em Trabalhos de Conclusão de Curso, reduzindo a necessidade de revisões manuais*, foi adequadamente respondida por meio do desenvolvimento e da avaliação do sistema ARAMIS. Os resultados obtidos demonstram que a arquitetura multiagente integrada a um LLM *open-source* permite a geração de revisões consistentes, cientificamente fundamentadas e direcionadas a aspectos centrais da escrita acadêmica. O ARAMIS mostrou-se capaz de identificar incoerências textuais, fragilidades metodológicas e problemas de encadeamento lógico, oferecendo comentários precisos que preservam o estilo autoral do estudante, ao mesmo tempo em que contribuem para o aumento do rigor metodológico e da correção gramatical do texto. Assim, a ferramenta se consolida como um assistente acadêmico eficaz, atuando como suporte qualificado ao processo de revisão e reduzindo a dependência de intervenções manuais recorrentes por parte dos orientadores.

Os resultados obtidos por meio do *score* SUS (90,5) confirmam que a ferramenta atende às expectativas de usabilidade de estudantes de graduação, contribuindo para a redução do tempo dedicado a revisões de caráter preliminar e para a melhoria das versões iniciais dos trabalhos acadêmicos. Em suma, este trabalho valida o uso de modelos de linguagem generativa em tarefas de especialização linguística, reforçando a viabilidade de soluções *open-source* e contribuindo para a democratização do acesso a tecnologias de suporte à escrita científica no contexto brasileiro.

Para trabalhos futuros, sugere-se o uso do *fine-tuning* em um modelo *open-source* para auxiliar na correção de trabalhos científicos em língua portuguesa. Além de:

1. Ampliar a base de usuários para efetuar a avaliação da usabilidade da ferramenta, além de utilizar outras métricas específicas para medir a precisão das revisões geradas pelo LLM;
2. Aplicação de técnicas de recuperação de informação como o *Retrieval-Augmented Generation* (RAG) para que as revisões retornadas pelo modelo tenham uma especificidade mais elevada e confiança maior sobre o assunto que está tratando;
3. Melhorar o fluxo de autenticação e concorrência no processamento de dados da ferramenta, permitindo que uma quantidade maior de usuários acesse o sistema sem que haja inconsistências;
4. Hospedagem em um servidor que suporte um modelo de médio porte como o *gpt-oss-20b*, e que suporte o volume de dados demandado;
5. Aprimorar a interface gráfica do sistema, aplicando tecnologias modernas de interface de usuário que melhorem a experiência de uso e a usabilidade;

REFERÊNCIAS

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

BANGOR, A.; KORTUM, P.; MILLER, J. Determining what individual sus scores mean: Adding an adjective rating scale. **Journal of usability studies**, Usability Professionals' Association Bloomingdale, IL, v. 4, n. 3, p. 114–123, 2009.

BROOKE, J. Sus: A quick and dirty usability scale. **Usability Eval. Ind.**, v. 189, 11 1995.

CARBONI, R. M.; NOGUEIRA, V. d. O. Facilidades e dificuldades na elaboração de trabalhos de conclusão de curso. **ConScientiae Saúde**, v. 3, p. 65–72, jan. 2008. Disponível em: https://periodicos.uninove.br/saude/article/view/321.

CHAMOUN, E.; SCHLICHTKRULL, M.; VLACHOS, A. Automated focused feedback generation for scientific writing assistance. In: KU, L.-W.; MARTINS, A.; SRIKUMAR, V. (Ed.). **Findings of the Association for Computational Linguistics: ACL 2024**. Bangkok, Thailand: Association for Computational Linguistics, 2024. p. 9742–9763. Disponível em: https://aclanthology.org/2024.findings-acl.580/.

CHOWDHARY, K. R. Natural language processing. In: ____. **Fundamentals of Artificial Intelligence**. New Delhi: Springer India, 2020. p. 603–649. ISBN 978-81-322-3972-7. Disponível em: https://doi.org/10.1007/978-81-322-3972-7_19.

COPPIN, B. **Artificial Intelligence Illuminated**. [*S. l.*]: Jones and Bartlett Publishers, 2004.

D'ARCY, M.; HOPE, T.; BIRNBAUM, L.; DOWNEY, D. **MARG: Multi-Agent Review Generation for Scientific Papers**. 2024. Disponível em: https://arxiv.org/abs/2401.04259.

DIAS, D. d. S.; SILVA, M. F. d. **Como escrever uma monografia**. [*S. l.*]: Universidade Federal do Rio de Janeiro, 2009.

FUCHS, K. Exploring the opportunities and challenges of nlp models in higher education: is chat gpt a blessing or a curse? **Frontiers in Education**, Volume 8 - 2023, 2023. ISSN 2504-284X. Disponível em: https://www.frontiersin.org/journals/education/articles/10.3389/feduc.2023.1166682.

HESTON, T. F.; KHUN, C. Prompt engineering in medical education. **International Medical Education**, MDPI, v. 2, n. 3, p. 198–205, 2023.

IDAHL, M.; AHMADI, Z. OpenReviewer: A specialized large language model for generating critical scientific paper reviews. In: DZIRI, N.; REN, S. X.; DIAO, S. (Ed.). **Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (System Demonstrations)**. Albuquerque, New Mexico: Association for Computational Linguistics, 2025. p. 550–562. ISBN 979-8-89176-191-9. Disponível em: https://aclanthology.org/2025.naacl-demo.44/.

JOHRI, P.; KHATRI, S. K.; AL-TAANI, A.; SABHARWAL, M.; SUVANOV, S.; CHAUHAN, A. Natural language processing: History, evolution, application, and future work. In: ____. [*S. l.*: *s. n.*], 2021. p. 365–375. ISBN 978-981-15-9711-4.

JOSHI, A.; KALE, S.; CHANDEL, S.; PAL, D. K. Likert scale: Explored and explained. **British journal of applied science & technology**, Sciencedomain International, v. 7, n. 4, p. 396, 2015.

LIANG, W.; ZHANG, Y.; CAO, H.; WANG, B.; DING, D. Y.; YANG, X.; VODRAHALLI, K.; HE, S.; SMITH, D. S.; YIN, Y. *et al.* Can large language models provide useful feedback on research papers? a large-scale empirical analysis. **NEJM AI**, Massachusetts Medical Society, v. 1, n. 8, p. AIoa2400196, 2024.

LUNSFORD, A. A.; LUNSFORD, K. J. "mistakes are a fact of life": A national comparative study. **College Composition and Communication**, National Council of Teachers of English, v. 59, n. 4, p. 781–806, 2008. ISSN 0010096X. Disponível em: http://www.jstor.org/stable/20457033.

MARVIN, G.; HELLEN, N.; JJINGO, D.; NAKATUMBA-NABENDE, J. Prompt engineering in large language models. In: SPRINGER. **International conference on data intelligence and cognitive informatics**. [*S. l.*], 2023. p. 387–402.

NADKARNI, P. M.; OHNO-MACHADO, L.; CHAPMAN, W. W. Natural language processing: an introduction. **Journal of the American Medical Informatics Association**, BMJ Group BMA House, Tavistock Square, London, WC1H 9JR, v. 18, n. 5, p. 544–551, 2011.

NAVEED, H.; KHAN, A. U.; QIU, S.; SAQIB, M.; ANWAR, S.; USMAN, M.; AKHTAR, N.; BARNES, N.; MIAN, A. A comprehensive overview of large language models. **ACM Transactions on Intelligent Systems and Technology**, ACM New York, NY, 2023.

OECD. **PISA 2022 Results (Volume I): The State of Learning and Equity in Education**. Paris, 2023. Disponível em: https://doi.org/10.1787/53f23881-en.

PADRINI-ANDRADE, L.; BALDA, R. d. C. X.; ARECO, K. C. N.; BANDIERA-PAIVA, P.; NUNES, M. d. V.; MARBA, S. T. M.; CARVALHO, W. B. d.; RUGOLO, L. M. S. d. S.; ALMEIDA, J. H. C. d.; PROCIANOY, R. S. *et al.* Evaluation of usability of a neonatal health information system according to the user's perception. **Revista Paulista de Pediatria**, Sociedade de Pediatria de São Paulo, v. 37, n. 1, p. 90–96, Jan 2019. ISSN 0103-0582. Disponível em: https://doi.org/10.1590/1984-0462/;2019;37;1;00019.

PEREZ, M. S.; BOENAVIDES, W. M. Os limites para a revisão do texto literário a partir dos conceitos de autoria e estilo de bakhtin. **Bakhtiniana: Revista de Estudos do Discurso**, LAEL/PUC-SP (Programa de Estudos Pós-Graduados em Linguística Aplicada e Estudos da Linguagem da Pontifícia Universidade Católica de São Paulo), v. 12, n. 1, p. 113–130, Jan 2017. ISSN 2176-4573. Disponível em: https://doi.org/10.1590/2176-457325830.

QIN, L.; CHEN, Q.; FENG, X.; WU, Y.; ZHANG, Y.; LI, Y.; LI, M.; CHE, W.; YU, P. **Large Language Models Meet NLP: A Survey**. 2024.

RAMOS, I. M. L. Orientações gerais do tcc. **São Paulo (SP): Centro Paula Souza**, 2011.

REYNOLDS, L.; MCDONELL, K. Prompt programming for large language models: Beyond the few-shot paradigm. **CoRR**, abs/2102.07350, 2021. Disponível em: https://arxiv.org/abs/2102.07350.

SALOMON, D. V. **Como fazer uma monografia**. [*S. l.*]: Martins Fontes, 2004.

SILVA, D. F. da; FOGGIATO, A. A.; NETO, J. L. T.; PARREIRAS, S. O. **Manual prático para elaboração de trabalhos de conclusão de curso**. [*S. l.*]: Editora Blucher, 2020.

SRIVARSHA, N.; NITHIN, G.; HARIHARAN, S.; DASH, B. B.; CHOWDHURY, S.; PATRA, S. S. Auto text correction using nlp techniques. In: **2025 6th International Conference on Mobile Computing and Sustainable Informatics (ICMCSI)**. [*S. l.*: *s. n.*], 2025. p. 590–594.

THIRUNAVUKARASU, A. J.; TING, D. S. J.; ELANGOVAN, K.; GUTIERREZ, L.; TAN, T. F.; TING, D. S. W. Large language models in medicine. **Nature medicine**, Nature Publishing Group US New York, v. 29, n. 8, p. 1930–1940, 2023.

WEI, J.; WANG, X.; SCHUURMANS, D.; BOSMA, M.; XIA, F.; CHI, E.; LE, Q. V.; ZHOU, D. *et al.* Chain-of-thought prompting elicits reasoning in large language models. **Advances in neural information processing systems**, v. 35, p. 24824–24837, 2022.

ZHANG, M.; ZHANG, Y. The study on nlp-based semantic analysis technology to improve the accuracy of english translation. In: **2023 IEEE International Conference on Control, Electronics and Computer Technology (ICCECT)**. [*S. l.*: *s. n.*], 2023. p. 1107–1112.

ZHAO, W. X.; ZHOU, K.; LI, J.; TANG, T.; WANG, X.; HOU, Y.; MIN, Y.; ZHANG, B.; ZHANG, J.; DONG, Z. *et al.* A survey of large language models. **arXiv preprint arXiv:2303.18223**, v. 1, n. 2, 2023.