MarIA: Um sistema de apoio a RSL baseado em Inteligência Artificial

Gustavo Corrêa Rodrigues[1], **Isadora Mendes dos Santos**[2], **Marcelle Pereira Mota**[1,2]

[1] Faculdade de Computação
Universidade Federal do Pará (UFPA), Belém, PA - Brasil

[2]Programa de Pós Graduação em Computação
Universidade Federal do Pará (UFPA), Belém, PA - Brasil

{gustavo.rodrigues,isadora.mendes}@icen.ufpa.br, mpmota@ufpa.br

Abstract. *Systematic literature reviews (SLR) are studies carried out to assess the scientific literature about certain topic via a rigorous method. One of the steps of such studies is abstract screening, where the researcher evaluates from hundreds to thousands titles and abstracts regarding their relevance to the research. Here, there's an opportunity to use artificial intelligence to improve the efficiency of this process. This papers explores the use of generative artificial intelligence this process via the implementation and testing of a abstract screening tool assisted by an AI chatbot. The results of this paper points that tools like the one presented are viable and can better the experience of the researcher that executes an SLR.*

Resumo. *Revisões sistemáticas da literatura (RSL) são estudos realizados para avaliar a literatura cientifica sobre um determinado tópico por meio de um método rigoroso. Uma das etapas destes estudos é a seleção de artigos, onde o pesquisador avalia de centenas a milhares de artigos quanto à relevância destes para a pesquisa. Aqui, existe a oportunidade do uso da inteligência artificial para melhorar a eficiência deste processo. Este trabalho explora o uso de inteligência artificial generativa neste processo através da implementação e testes de uma ferramenta de seleção de artigos auxiliada por um chatbot de IA. Os resultados deste estudo apontam que ferramentas como a apresentada são viáveis e podem melhorar a experiência do pesquisador que executa uma RSL.*

1. Introdução

Revisões sistemáticas da literatura (RSLs) são estudos que avaliam e interpretam a literatura existente sobre um determinado tópico em busca de respostas por meio de um método rigoroso, confiável e reproduzível. Os métodos e procedimentos para a execução deste tipo de estudo são estabelecidos e consagrados [Kitchenham et al. 2007]. Uma das etapas para a realização deste tipo estudo é a seleção de artigos, nesta etapa, o pesquisador irá avaliar os títulos e resumos de potencialmente centenas a milhares de artigos procurando trabalhos relevantes para o tema da RSL, um processo repetitivo e cansativo.

Dado esse contexto, há uma oportunidade de usar ferramentas de Inteligência Artificial (IA) para melhorar a experiência e eficiência do pesquisador [van Dijk et al. 2023, de la Torre-López et al. 2023]. O objetivo deste trabalho é explorar a aplicação de inteligência artificial generativa (IAG), especificamente modelos de linguagem em larga

escala (LLMs), para a execução de RSL através da implementação de um sistema para a seleção de artigos alimentado por inteligência artificial e depois executar testes para estudar a viabilidade de um sistema como este.

A motivação para a execução deste trabalho surgiu, principalmente, da observação direta dos desafios enfrentados por pesquisadores na etapa de seleção de artigos em RSL por meios tradicionais, como planilhas. Além disso, apesar de ferramentas semelhantes com a que é proposta neste artigo existirem, elas tendem a ter um escopo mais aberto, sendo voltadas para tarefas de produtividade em geral, poucas se encontram nesta entre ferramentas de inteligência artificial e ferramentas para RSL.

Dado o exposto, este trabalho propõe e apresenta nas seções a seguir o protótipo de um sistema, MarIA (**Ma**peamentos e **r**evisões com **IA**), que oferece uma maneira de selecionar artigos em uma RSL auxiliada por IA, permitindo que o pesquisador consulte um chatbot que responde perguntas considerando os artigos importados para a RSL. A Seção 2 apresenta ferramentas e trabalhos já publicados na literatura relacionados a este, a Seção 3 apresenta a Metodologia tomada no desenvolvimento de MarIA, a Seção 5 apresenta o sistema em si, a Seção 6 apresenta os resultados do teste da ferramenta, a Seção 4 apresenta a arquitetura do sistema, a Seção 7 apresenta uma discussão acerca do trabalho e por fim, a Seção 8 apresenta as considerações finais, limitações deste trabalho, bem como trabalhos futuros.

2. Trabalhos e Ferramentas Relacionadas

Porifera [Campos et al. 2022] é um sistema web e colaborativo para a execução de revisões sistemáticas da literatura e estudos de mapeamento sistemático. A proposta de Porifera é a colaboração, permite o registro do parecer de todos os pesquisadores sobre qualquer artigo e, em caso de discordância, permite o registro de uma decisão em consenso, porém mantendo as decisões originais, garantindo a transparência e a rastreabilidade, além disso, o sistema conta com métodos de verificação da confiabilidade da seleção, através do cálculo da concordância absoluta e do índice Kappa.

ASReview [van de Schoot et al. 2021] é um sistema de seleção de artigos em RSL mediado por aprendizado de máquina ativo, para o funcionamento correto da aplicação, ao carregar a coleção de artigos, pelo menos um artigo precisa ser aceito e um precisa ser rejeitado para o funcionamento correto da aplicação, após esse processo inicial, um modelo de aprendizado de máquina é treinado e um novo artigo é apresentado para o usuário para classificação, com essa nova classificação o modelo é treinado novamente e o ciclo se repete até todos os artigos serem classificados. Esse processo, em essência, reordena a lista de artigos a serem apresentados a cada classificação, fazendo com que os artigos com mais chances de serem aprovados sejam apresentados primeiro. Este método faz com que 95% dos artigos relevantes sejam encontrados ao classificar apenas 8 a 33% dos artigos totais.

Além destes, existem ferramentas com propostas semelhantes que não possuem trabalhos publicados, por exemplo, Parsifal [Parsifal 2024] é um software colaborativo para a realização de RSLs em times, a aplicação possui integração com motores de pesquisa de artigos como o Scopus e é otimizada para a colaboração entre pesquisadores. Parsifal é uma ferramenta para todas as etapas da realização da RSL, desde a fase de planejamento até a fase de condução.

Por se tratar de um protótipo, MarIA possui, comparadamente, menos funções que os sistemas apresentados acima, os quais são sistemas completos, apesar disso, é importante mencionar a diferentes propostas apresentadas. O ASReview toma as rédeas do processo de classificação, ativamente modificado a ordem que os artigos são apresentados para fazer com que os mais relevantes sejam vistos primeiro, já MarIA propõe que o pesquisador tome decisões apoiadas pelo uso de um chatbot e apresenta uma forma alternativa de interagir com os artigos sendo analisados. Enquanto isso, Porifera não possui aspectos de IA, sua proposta envolve fazer com que as classificações e decisões em revisões sendo executadas por mais de um pesquisador sejam tomadas em consenso.

3. Metodologia

O principal uso da aplicação é substituir o uso de planilhas para realizar a seleção de artigos em uma RSL e enriquecer esse processo com inteligência artificial. Para realizar a entrada de dados na ferramenta, optou-se por utilizar arquivos BibTeX [BibTeX 2006], devido ao fato destes arquivos poderem ser exportados da maioria de bases de busca de artigos e, caso sejam utilizadas múltiplas bases, poderem ser combinados em um só sem muitas complicações ou necessidade de ferramentas específicas, bastando um editor de texto simples. Os artigos importados na plataforma podem, então, serem exportados novamente, no formato BibTeX originalmente usado para importá-los ou em um arquivo CSV, para importação em outras ferramentas.

Devido à complexidade das operações necessárias para atingir os objetivos do sistema, MarIA utiliza a arquitetura cliente-servidor, um modelo de computação distribuída onde o um servidor centralizado e remoto, atende e processa requisições iniciadas nos vários clientes, executados no computador do usuário e responsáveis por iniciar requisições e mostrar seus resultados ao usuário por meio de uma interface gráfica de usuário (GUI). No caso de MarIA, por se tratar de uma aplicação web, o servidor é chamado de*backend* e os clientes são denominados *frontend*.

Em MarIA, o *backend* possui três responsabilidades. A primeira é realizar controle de acesso, garantindo que os usuários tenham acesso aos seus, e somente aos seus, artigos e mensagens com o chatbot, persistindo-os entre as sessões. A segunda é realizar o processamento dos dados de entrada e saída, bem como a atualização dos dados existentes, ou seja, converter os arquivos importados em estruturas de dados para manipulação no sistema, realizar as operações de aceite dos artigos e criar os arquivos para exportação. Por fim, a terceira é, fazendo o uso de tecnologias específicas para isso, fornecer o serviço de chatbot que será consumido pelos clientes.

Para a construção de um servidor capaz de fornecer os serviços citados, optou-se por usar o ambiente de execução NodeJs [NodeJS 2024], para executar código Javascript fora do ambiente do navegador. Essa decisão foi tomada não somente pelo vasto ecossistema de bibliotecas a frameworks construídos em Javascript mas também pelo fato do ambiente Node ser estabelecido na indústria e amplamente documentado.

3.1. Testes de viabilidade

Após a implementação do sistema, foi realizado um teste interno do sistema, com membros do grupo de pesquisa, o objetivo deste teste foi verificar a viabilidade do uso de MarIA para a execução de RSLs. Este teste consistia na simulação de uma RSL sobre o uso

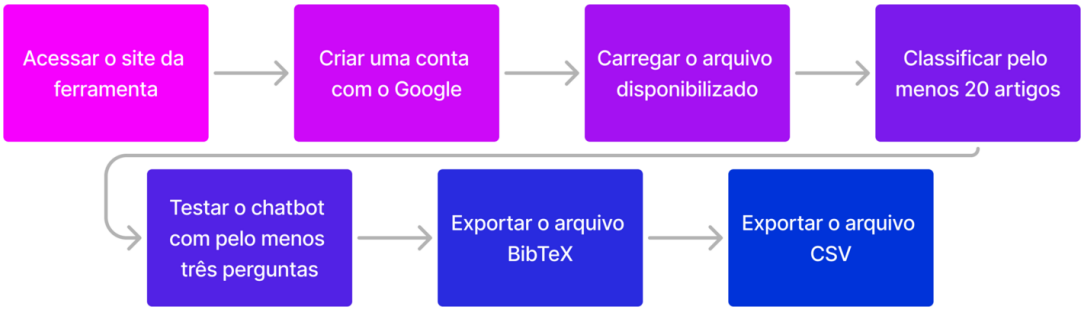

Figura 1. Roteiro de testes.

de realidade aumentada na educação, onde os voluntários usaram o resultado da busca realizada na *Association for Computing Machinery Digital Library* (ACM DL) "augmented reality" OR "AR" OR "mixed reality" OR "virtual reality") AND ("education" OR "learning experiences" OR "student engagement" OR "educational technology"), filtrando os resultados para serem retornados apenas artigos publicados na *Conference on Human Factors in Computing Systems* de 2024 (CHI 24). Com base nessa coleção de artigos, foi solicitado que os voluntários executassem o roteiro de testes que pode ser visto na Figura 1

Os passos acima foram elaborados para simular o fluxo de trabalho ao utilizar o MarIA, eles exploram todas as funcionalidades principais do sistema, nomeadamente, função de classificar os artigos, o chatbot, e a possibilidade de exportar a coleção de artigos em formatos úteis para o resto da condução da RSL. Os usuários acabaram utilizando a ferramenta por, em média, 30 minutos. Após a execução do roteiro de testes, foram realizadas entrevistas abertas com os voluntários, as perguntas foram elaboradas para obter feedback sobre a viabilidade e a experiência de uso do sistema, no total, três testes e entrevistas foram realizados, as entrevistas duraram 30 minutos cada, totalizando uma hora para execução do teste mais a entrevista. As perguntas realizadas para os usuários foram as segunites:

1. O que você achou do processo de seleção dos artigos?
2. Houve desafios encontrados durante o uso do MarIA que você sente que não aconteceriam caso estivesse usando outro método para a seleção de artigos? (planilhas, Parsifal, Porifera etc)
3. Cite duas características do sistema que você gostou, explique os motivos;
4. Cite duas características do sistema que você não gostou, explique os motivos;
5. Quais funções você adicionaria no sistema?
6. Como foi sua experiência usando o chatbot?
7. As respostas do chatbot foram relevantes considerando o contexto?
8. O chatbot atendeu suas expectativas?
9. Alguma das funções do sistema é irrelevante ou redundante? Se sim, qual?
10. Você usaria o sistema novamente?
11. Qual a probabilidade de você recomendar o MarIA para outro pesquisador?

4. Aspectos Técnicos

Um diagrama com a arquitetura do sistema pode ser encontrado na Figura 2.

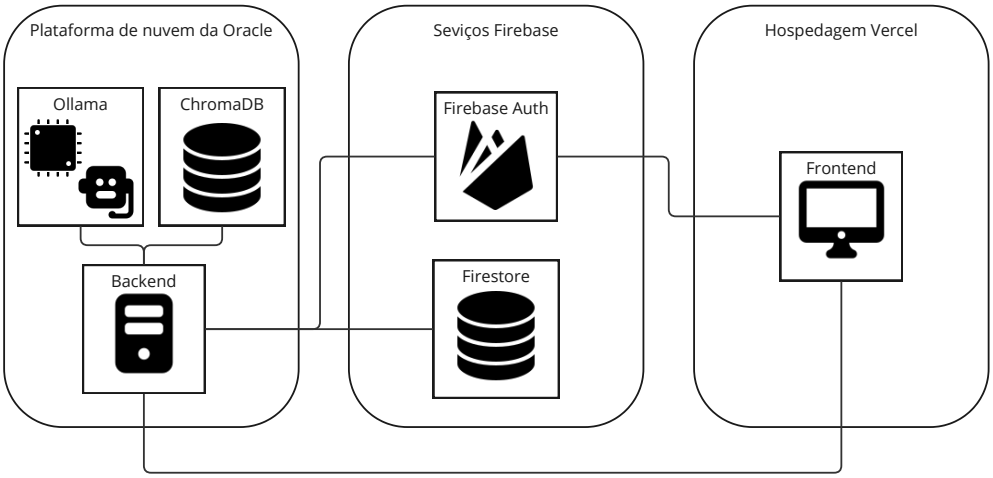

Figura 2. Arquitetura do Sistema.

Para a criação da interface, foi utilizado o *framework* Quasar [Quasar Framework 2024], que fornece componentes e funções auxiliares construídos com a biblioteca de interface *Vue* [VueJS 2024], que combina HTML, CSS e *TypeScript* em um único arquivo, resultando nos componentes que podem ser utilizados para construir a aplicação. Além dessas tecnologias base, foram usadas diversas bibliotecas para o desenvolvimento da interface:

- *Markdown-it* [Markdown it 2024]: Transforma o texto estruturado em formato *markdown* em texto HTML, na aplicação, é usado para exibir as respostas geradas pela IA com a estilização correta;
- *Validator* [Validator 2024]: Biblioteca de validação de formulários, usada para validar o formulário de carregamento de arquivo;
- *Multiparty* [Multiparty 2024]: Biblioteca que realiza automaticamente a codificação do formulário com arquivos para o envio ao *backend*;
- *Axios* [Axios 2024]: Biblioteca que encapsula as requisições HTTP/HTTPS em *promises*, facilitando o tratamento de erros e fluxo de execução do aplicativo;
- *Firebase* [Firebase 2024]: Plataforma de serviços em nuvem oferecida pelo Google, possui diversos serviços, porém, no *frontend* da aplicação apenas o módulo de autenticação foi utilizado;
- *Vuefire* [Vuefire 2024]: Biblioteca que adapta o Firebase para ser usado no Vue, encapsulando as funções do Firebase em funções globais (no caso da API de opções) e funções de composição (no caso da API de composição);
- *Bibtex-parser* [Bibtex parser 2024]: Biblioteca que converte *strings BibTeX* em objetos Javascript, no *frontend* foram utilizados apenas os tipos declarados na biblioteca;
- *Vue-i18n* [Vue i18n 2024]: Biblioteca de internacionalização adaptada para aplicações construídas com Vue, MarIA está disponível em inglês e português.

O *backend* é movido pela biblioteca *Express* [Express 2024], que permite a criação de servidores HTTP utilizando *NodeJS*. Para o desenvolvimento desse servidor, também

foi utilizado o TypeScript, além disso, para armazenar os dados foram utilizadas duas bases de dados diferentes, ChromaDB [ChromaDB 2024], uma base de dados vetorial, e a base de dados *Firestore* disponibilizada no *Firebase*, uma base de dados de documentos.

TypeScript é um superconjunto da linguagem de programação JavaScript. A principal diferença do TypeScript para o JavaScript tradicional é a presença de tipagem forte e estática, enquanto JavaScript é uma linguagem fraca e dinamicamente tipada, isso permite que o editor de texto encontre erros antes da execução do código, evitando bugs inesperados, além disso, TypeScript melhora a experiência de desenvolvimento, haja vista que, devido à presença de tipos, editores de texto podem oferecer *IntelliSense* mais completo.

Para o processo de autenticação, a aplicação recorre ao *Firebase*. Apesar da biblioteca permitir a criação de usuários próprios da aplicação, com login e senha, foi decidido usar apenas contas *Google* para realizar a autenticação, essa decisão foi tomada considerando aspectos de segurança, eliminando a necessidade de realizar o armazenamento dos logins e senhas dos usuários, além de evitar a necessidade de implementar uma estrutura de autenticação própria.

Devido aos aspectos de inteligência artificial da aplicação, é necessário o uso de um serviço de LLM, por motivos de custo de simplicidade, optamos por utilizar o software *Ollama* [Ollama 2024]. O software permite a execução de modelos de LLMs via linha de comando, além disso, Ollama também disponibiliza a opção de interagir com os LLMs via uma API muito semelhante à API da OpenAI, desse modo, é possível incorporar um chatbot na aplicação usando requisições HTTP. Como modelo de geração de texto, foi usado o modelo Phi-3 Mini e para *word embedding*, o nomic-embed-text, da Nomic AI.

O Modelo Phi-3 Mini, desenvolvido pela *Microsoft* e lançado em abril de 2024 [Bilenko 2024], é tecnicamente caracterizado como um modelo de linguagem pequeno (SLM, do inglês, *Small Language Model*), essa classificação se dá pelo fato do modelo ser menor que modelos mais populares. O modelo GPT-3.5 da OpenAI, por exemplo, possui 175 bilhões de parâmetros, enquanto o modelo utilizado possui apenas 3.8 bilhões, apesar desta diferença de tamanho, seu desempenho é comparável com modelos maiores [Abdin et al. 2024].

O modelo nomic-embed-text é um modelo de *word embedding*, esses modelos são utilizados para realizar o processo de mesmo nome, *word embedding* é uma técnica desenvolvida na área de Processamento de Linguagem Natural (PLN), uma sub-área da Inteligência Artificial, que consiste na transformação de palavras e textos em vetores numéricos que armazenam o contexto semântico das palavras, representando-as efetivamente com números. Uma propriedade deste processo que é essencial para o processo de RAG explorado neste trabalho é a capacidade destes vetores de permitir comparar a similaridade entre as palavras, ou seja, palavras semelhantes resultarão em vetores semelhantes. Outra propriedade importante destes vetores é o fato das relações entre as palavras serem consistentes, por exemplo, a distância no espaço vetorial entre palavras que são nomes de frutas e o nome das árvores onde essas frutas crescem são semelhantes para pares de frutas e árvores [Jurafsky and Martin 2000].

Além dessas ferramentas principais, algumas outras bibliotecas foram utilizadas para o desenvolvimento do *backend* e estão listadas a seguir:

- *Chalk* [Chalk 2024]: Biblioteca usada para colorir a saída de texto via terminal da

aplicação, usada para facilitar a leitura dos registros de execução;

- *Multer* [Multer 2024]: Biblioteca usada para decodificar e armazenar arquivos recebidos via requisições HTTP, no caso de MarIA, os arquivos são recebidos, decodificados, armazenados, lidos e deletados em seguida, ou seja, não há persistencia das coleções enviadas além do armazenamento nas bases de dados;
- *Bibtex-parser*: Mesma biblioteca usada no *frontend*, porém aqui é usada pela sua principal funcionalidade, que é transformar texto *bibtex* em objetos javascript;
- *Cors* [Cors 2024]: Biblioteca usada para configurar o comportamento do CORS na aplicação;
- *Dotenv* [Dotenv 2024]: Utilizada para configurar as variáveis de ambiente na execução do servidor, principalmente se usado fora de um conteiner *docker*, haja vista que, nesse caso, as variáveis de ambiente são configuradas no comando *docker run* ou no arquivo de configuração do *compose*;
- *Axios*: Também utilizada no *frontend*, aqui é utilizada para realizar as requisições à API disponibilizada pelo *ollama*.

Quando um arquivo é carregado no *frontend*, ele é enviado via HTTP para o *backend* para processamento e armazenamento, o arquivo é processado usando a biblioteca *bibtex-parser* e o resultado é enviado ao *frontend* para exibição, esse resultado é armazenado em uma coleção do *ChromaDB*, uma base de dados vetorial, ou seja, indexa os documentos com um vetor de números que, nesse caso, é gerado pelo modelo de *embedding*. Esse processo pode demorar, dependendo do tamanho do arquivo inserido, por isso, os arquivos também são armazenados no *Firestore*, permitindo que o usuário saia imediatamente da aplicação sem perder dados. Após o armazenamento no *ChromaDB*, os dados são deletados do *Firestore*.

Quando uma mensagem (*query*) é enviada ao *chatbot*, ela passa pelo mesmo processo de *word embedding* dos documentos carregados pelo arquivo BibTeX, esse processo permite que a base de dados consiga retornar artigos relacionados ao tema da pergunta, esses artigos são adicionados à janela de contexto do LLM para a geração da resposta. Esse processo é chamado de Geração aumentada por Recuperação (*Retrieval-Augmented Generation*, RAG), na Figura 7 é possível encontrar um fluxograma do processo de RAG.

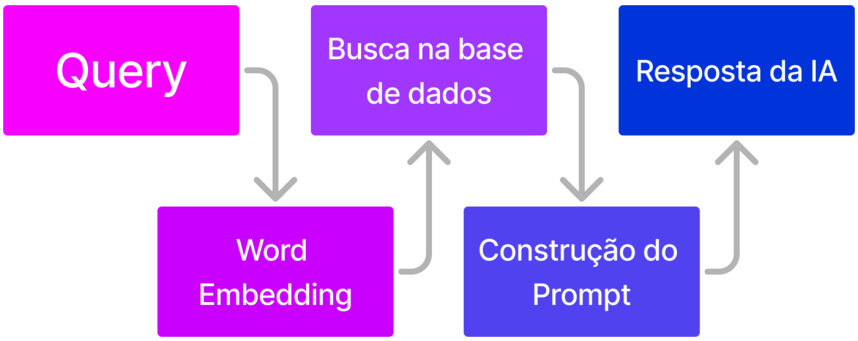

Figura 3. Processo de RAG.

O código-fonte de MarIA está armazenado em repositório Git hospedado na plataforma Github [Github 2024], o cliente se encontra hospedado na plataforma de hospedagem Vercel [Vercel 2024]. O servidor encontra-se hospedado na plataforma de nuvem da Oracle [Oracle 2024] (tanto a base de dados, o serviço de LLM e o servidor em si) e é servido utilizando a ferramenta de conteinerização Docker [Docker 2024].

5. Apresentando: MarIA

A primeira tela ao abrir a aplicação é uma tela inicial simples, contendo uma breve apresentação sobre a aplicação e um botão para prosseguir, ao pressionar o botão, os usuários são redirecionados ao fluxo de login externo do *Google*, MarIA usa contas do *Google* para autenticação e não possui login e senha própios. Usuários não autenticados não conseguem usar o sistema, essa decisão foi tomada devido à necessidade de um id de usuário único (disponibilizado pela autenticação) para armazenar as coleções de artigos dos diferentes usuários. Na Figura 4 é possível encontrar uma captura desta tela.

Figura 4. Tela inicial de MarIA.

Ao fazer login, o usuário é redirecionado para a principal tela da aplicação, onde ele poderá interagir com o sistema de fato, essa tela é simples e contém apenas dois campos de formulário, em um deles, o usuário pode digitar palavras-chave e no outro, pode inserir um arquivo para importar artigos na base de dados. Ao importar um arquivo, o sistema recupera os dados dos artigos e os exibe para o usuário em forma de cartões contendo o título, autores, resumo e palavras-chave do artigo, o título do artigo é um link para visitar a página do artigo na internet. Essa tela pode ser visualizada na Figura 5

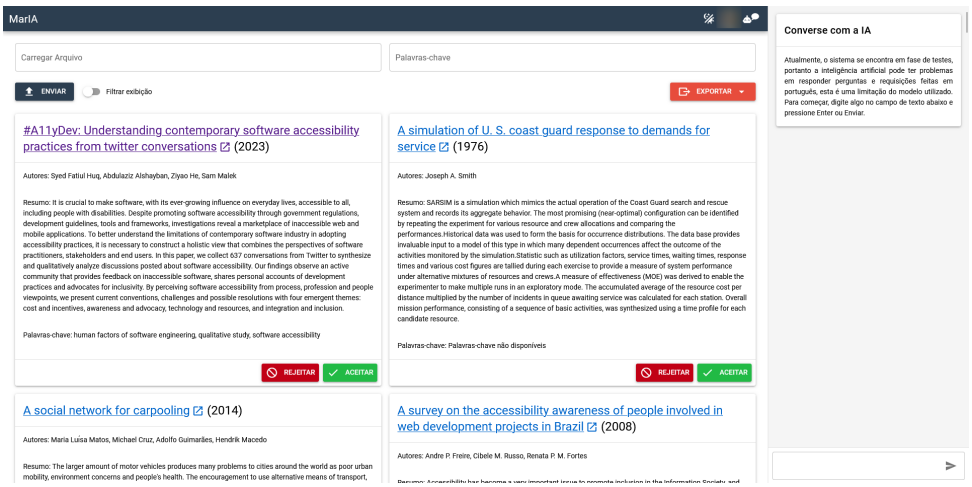

Figura 5. Tela principal de MarIA.

Com os artigos importados, o usuário pode usar os botões disponíveis em cada cartão para aceitar ou rejeitá-los, esse processo pode ser realizado ao longo de múltiplas seções, pois os dados são armazenados, além disso, ao exportar os artigos como CSV, os artigos terão seu estado mantido, é possível ver na figura 6 exemplos de artigos que ja passaram por este processo na ferramenta. Com uma base de artigos importada, o assistente de inteligência artificial pode começar a responder perguntas e requisições com informações relevantes. Por se tratar de um protótipo com recursos computacionais limitados, atualmente, não é possível solicitar que a inteligência artificial interaja com os artigos, por exemplo, requisições como "Rejeite todos os artigos que..." não produzem resultado, um exemplo de mensagem produzida pelo chatbot pode ser encontrado na Figura 7.

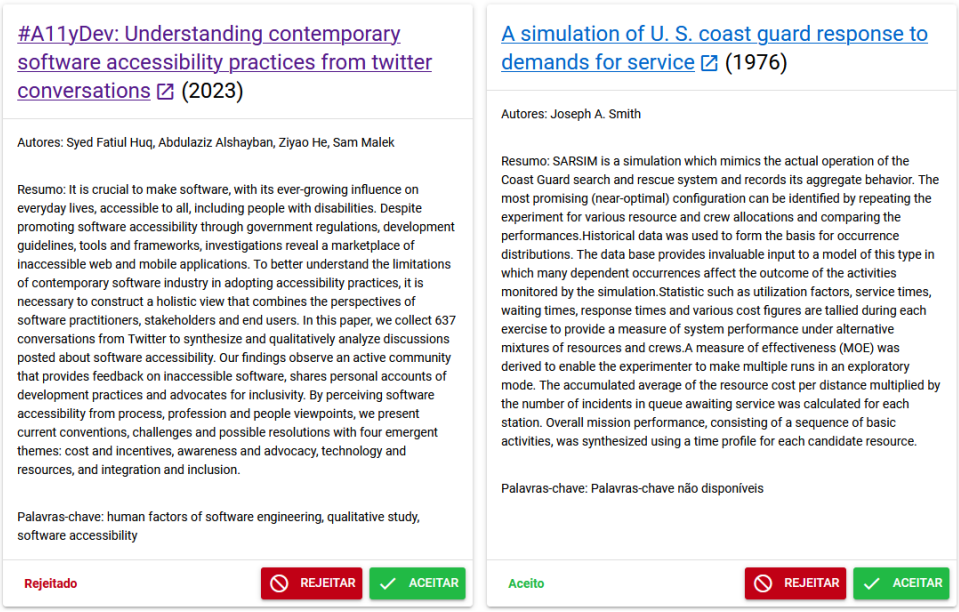

Figura 6. Artigos classificados.

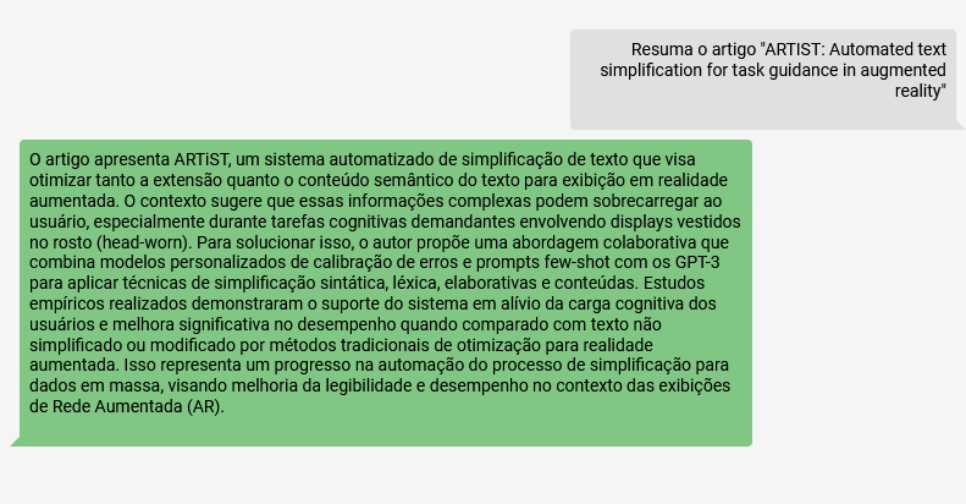

Figura 7. Mensagem elaborada pelo chatbot.

6. Resultados

Todos os testes realizados obtiveram sucesso quanto as realizações dos passos do roteiro, os três voluntários relataram ter gostado do processo de seleção de artigos, porém com algumas ressalvas, um dos voluntários mencionou sentir falta de uma descrição mais detalhada do sistema na página inicial e sentiu dificuldades para dar início no processo, entretanto, uma vez iniciado, o resto do teste correu bem. Quanto aos possíveis desafios encontrados durante o processo de seleção, apenas um voluntário relatou ter sentido dificuldades a mais, especificamente por achar que a interface atual do sistema pode causar uma sensação de "muito trabalho pela frente", devido à disposição dos cartões representando os artigos uma vez que a coleção é importada no sistema.

Quanto as características positivas do sistema, dois dos três voluntários mencionaram terem gostado do chatbot, como pontos positivos, citaram o fato do chatbot permitir encontrar artigos relevantes para a pesquisa sem que eles tenham algum termo chave, porém estejam alinhados ao contexto, e o fato do chatbot permitir que o pesquisador sane possíveis dúvidas de maneira rápida enquanto espera a resposta de um par. Outro ponto mencionado como positivo por dois voluntários foi a exportação dos artigos em diferentes formatos, por ser congruente com a proposta do sistema e pelos arquivos exportados refletirem a seleção dos artigos. Além disso, os voluntários mencionaram como pontos positivos a interface minimalista e a busca por palavras-chave.

Já em relação aos pontos negativos, dois voluntários mencionaram que a disposição do formulário de envio de arquivo e inserção de palavras-chave gerou confusão, durante a execução do teste, os três voluntários clicaram no botão "enviar" antes de selecionar o arquivo para *upload*, além disso, os voluntários sentiram dificuldade em associar o botão de filtrar exibição com o campo de palavras-chave devido a sua distância no formulário. Um dos voluntários mencionou o fato dos artigos permanecerem na mesma ordem ao serem classificados como um ponto negativo, o voluntário disse que se sentiria mais confortável caso os artigos classificados fossem para o fim da lista exibida pelo sistema, agilizando o processo de revisão.

Em resposta à pergunta "Quais funcionalidades você adicionaria no sistema?", os voluntários mencionaram as seguintes funções:

- Opção de visualizar os arquivos exportados antes do download;
- Contador de artigos aprovados e rejeitados;
- Uma terceira opção de classificação, para expressar dúvida
- Métodos alternativos de ordenação dos artigos (atualmente os artigos são exibidos em ordem alfabética)
- Opção de visualizar os artigos em formato de lista expansível;
- Opção de traduzir os títulos e os resumos sem deixar a plataforma.

Todos os voluntários relataram ter tido uma experiência positiva com o chatbot, que atendeu as expectativas e, na maioria das vezes, respondeu às requisições com respostas relevantes para o contexto da pesquisa, nesse caso, durante um dos testes, após o voluntário solicitar que o chatbot resumisse um dos artigos, o chatbot respondeu com informações que não estavam presentes no corpus de artigos.

Finalmente, apenas um dos voluntários achou funções redundantes no sistema, especificamente a função de busca por palavras-chave, que, conforme o voluntário, o

chatbot também pode realizar esse tipo de busca, de maneira até melhor. Todos os voluntários disseram que usariam o sistema novamente e que recomendariam o sistema para outros pesquisadores, apesar das pequenas intercorrências encontradas durante o teste.

7. Discussão

Após o lançamento e popularização do ChatGPT em 2022, ferramentas que incorporam inteligência artificial se tornaram abundantes e extremamente populares, ferramentas como o Microsoft Copilot, o chatbot Gemini ou GitHub Copilot são alguns exemplos de aplicações que incorporam LLMs em tarefas de produtividade, o primeiro permite que o usuário realize perguntas respondidas conforme o conteúdo da página aberta no navegador, o segundo é um chatbot que pode pesquisar na internet e o terceiro é uma ferramenta geradora de código integrada ao editor de texto VSCode.

Dentre estas diversas aplicações, é possível encontrar ferramentas que facilitam a execução de pesquisas cientificas. Ferramentas como o ChatPDF [ChatPDF 2024], que permite que o usuário "converse" com um documento PDF, o Research Rabbit [RabbitAi 2024], que consegue encontrar artigos semelhantes e relacionados dado uma coleção de artigos inseridas na aplicação. Além deste tipo de ferramenta, exitem também de assistência de escrita, como o Grammarly ou o próprio ChatGPT.

Nesse contexto, MarIA encontra-se numa posição que mistura características de ferramentas como o ChatPDF e Zotero, a habilidade de "conversar" com a base de artigos importada do motor de buscas permite encontrar relações entre os artigos que poderiam passar despercebidas num processo de seleção tradicional. Ferramentas de inteligência artificial como o ChatGPT mostraram potencial para a auxiliar a execução de pesquisas cientificas, MarIA oferece a possibilidade de utilizar um chatbot de IA enriquecido com o contexto da pesquisa sendo realizada.

O processo de seleção dos estudos para uma RSL é um processo repetitivo e muitas vezes cansativo, sistemas como MarIA podem facilitar a execução desse processo com ferramentas que o facilitam, mas ainda requerem a execução por um pesquisador, portanto, apesar de envolver IA, MarIA não realiza o trabalho pelo pesquisador, apenas o auxilia e oferece uma maneira alternativa de realizar a análise dos artigos em uma RSL.

8. Considerações Finais

Este artigo apresentou um protótipo da aplicação MarIA, desenvolvida para auxiliar a etapa de seleção de artigos em uma RSL usando inteligência artificial. A aplicação disponibiliza um chatbot que, através do processo de RAG, responde às perguntas do usuário conforme o contexto dos artigos inseridos na ferramenta. Como protótipo, a ferramenta apresentou resultados satisfatórios, conforme evidenciado pelos testes realizados, substituindo com sucesso o uso de planilhas e respondendo às requisições a IA considerando os artigos importados quando necessário.

A principal limitação de MarIA é atualmente o fato de ser apenas um protótipo, apesar disso, já é possível ver como tal ferramenta pode impactar a execução de uma revisão sistemática, permitindo que o pesquisador, por meio da interação com o chatbot, encontre relações entre os artigos que poderiam passar despercebidas. Outra limitação do sistema, mencionada na Seção 7 é a falta de processamento por GPUs, devido aos preços

proibitivos desse tipo de plataforma, optou-se por utilizar-se uma plataforma de cloud gratuita que realiza o processamento apenas em CPU fazendo com que o tempo entre a requisição e a obtenção da resposta seja elevado.

Um dos principais desafios durante a implementação desse sistema ocorreu na a fase de desenvolvimento e consistiu na elaboração de uma arquitetura que oferecesse resultados aceitáveis, considerando tanto a qualidade das respostas quanto o tempo de resposta, MarIA é, atualmente, gratuito, sem anúncios e sem limitações de uso, usuários cadastrados podem carregar uma quantidade ilimitada de artigos e realizar requisições ilimitadas para o chatbot, isso é possível devido ao fato do LLM estar hospedado em um servidor gratuito, ao invés de usar algum serviço oferecido por uma empresa, como a OpenAI ou a Mixtral.

Como trabalhos futuros, pretende-se atualizar o sistema, adicionando funcionalidades, tornando-o mais completo, por exemplo, a possibilidade de um pesquisador possuir mais de uma coleção de artigos (atualmente, a única maneira de se fazer isso é exportando e importando os diferentes arquivos BibTeX), a possibilidade de adicionar etiquetas aos artigos, de recuperar as citações BibTeX de cada artigo de maneira individual, funcionalidades de colaboração remota, fazer com que a aplicação destaque os artigos utilizados para gerar a resposta da IA, implementar a possibilidade de carregar arquivos PDF inteiros, tornando a ferramenta útil para a etapa de extração de dados, também, adicionar a opção do usuário escolher qual modelo utilizar, dentre outras funcionalidades comuns em softwares modernos. Quanto aos testes com a plataforma, pretende-se realizar uma avaliação dos modelos de Inteligência Artificial utilizados na construção do sistema, com o objetivo de testar a acurácia das informações retornadas pelo modelo de linguagem e das *embeddings* geradas pelo modelo de *text embedding*.

Referências

Abdin, M., Jacobs, S. A., Awan, A. A., Aneja, J., Awadallah, A., Awadalla, H., Bach, N., Bahree, A., Bakhtiari, A., Bao, J., Behl, H., Benhaim, A., Bilenko, M., Bjorck, J., Bubeck, S., Cai, Q., Cai, M., Mendes, C. C. T., Chen, W., Chaudhary, V., Chen, D., Chen, D., Chen, Y.-C., Chen, Y.-L., Chopra, P., Dai, X., Del Giorno, A., de Rosa, G., Dixon, M., Eldan, R., Fragoso, V., Iter, D., Gao, M., Gao, M., Gao, J., Garg, A., Goswami, A., Gunasekar, S., Haider, E., Hao, J., Hewett, R. J., Huynh, J., Javaheripi, M., Jin, X., Kauffmann, P., Karampatziakis, N., Kim, D., Khademi, M., Kurilenko, L., Lee, J. R., Lee, Y. T., Li, Y., Li, Y., Liang, C., Liden, L., Liu, C., Liu, M., Liu, W., Lin, E., Lin, Z., Luo, C., Madan, P., Mazzola, M., Mitra, A., Modi, H., Nguyen, A., Norick, B., Patra, B., Perez-Becker, D., Portet, T., Pryzant, R., Qin, H., Radmilac, M., Rosset, C., Roy, S., Ruwase, O., Saarikivi, O., Saied, A., Salim, A., Santacroce, M., Shah, S., Shang, N., Sharma, H., Shukla, S., Song, X., Tanaka, M., Tupini, A., Wang, X., Wang, L., Wang, C., Wang, Y., Ward, R., Wang, G., Witte, P., Wu, H., Wyatt, M., Xiao, B., Xu, C., Xu, J., Xu, W., Yadav, S., Yang, F., Yang, J., Yang, Z., Yang, Y., Yu, D., Yuan, L., Zhang, C., Zhang, C., Zhang, J., Zhang, L. L., Zhang, Y., Zhang, Y., Zhang, Y., and Zhou, X. (2024). Phi-3 technical report: A highly capable language model locally on your phone.

Axios (2024). Disponível em https://axios-http.com/ptbr/docs/intro/. Acesso em Julho de 2024.

BibTeX (2006). Disponível em https://www.bibtex.org/. Acesso em Julho de 2024.

Bibtex parser (2024). Disponível em https://www.npmjs.com/package/@retorquere/bibtex-parser/v/8.0.12. Acesso em Julho de 2024.

Bilenko, M. (2024). Introducing phi-3: Redefining what's possible with slms. Disponível em https://azure.microsoft.com/en-us/blog/introducing-phi-3-redefining-whats-possible-with-slms/. Acesso em Maio de 2024.

Campos, T. P. D., Damasceno, E. F., and Valentim, N. M. C. (2022). Porifera: A collaborative tool to support systematic literature review and systematic mapping study. In *Proceedings of the XXXVI Brazilian Symposium on Software Engineering*, SBES '22, page 452–457, New York, NY, USA. Association for Computing Machinery.

Chalk (2024). Disponível em https://www.npmjs.com/package/chalk/. Acesso em Maio de 2024.

ChatPDF (2024). Disponível em https://www.chatpdf.com/. Acesso em Maio de 2024.

ChromaDB (2024). Disponível em https://www.trychroma.com/. Acesso em Julho de 2024.

Cors (2024). Disponível em https://www.npmjs.com/package/cors/. Acesso em Maio de 2024.

de la Torre-López, J., Ramírez, A., and Romero, J. R. (2023). Artificial intelligence to automate the systematic review of scientific literature. *Computing*, 105(10):2171–2194.

Docker (2024). Disponível em https://www.docker.com/. Acesso em Junho de 2024.

Dotenv (2024). Disponível em https://www.npmjs.com/package/dotenv/. Acesso em Maio de 2024.

Express (2024). Disponível em https://expressjs.com/pt-br/. Acesso em Julho de 2024.

Firebase (2024). Disponível em https://firebase.google.com/. Acesso em Julho de 2024.

Github (2024). Disponível em https://github.com/human-interaction-with-tecnologies/MarIA. Acesso em Julho de 2024.

Jurafsky, D. and Martin, J. H. (2000). *Speech and language processing*. Prentice Hall series in artificial intelligence. Pearson, Upper Saddle River, NJ. Capítulo 6.

Kitchenham, B., Charters, S., et al. (2007). Guidelines for performing systematic literature reviews in software engineering.

Markdown it (2024). Disponível em https://github.com/markdown-it/markdown-it. Acesso em Julho de 2024.

Multer (2024). Disponível em https://www.npmjs.com/package/multer/. Acesso em Maio de 2024.

Multiparty (2024). Disponível em https://www.npmjs.com/package/multiparty/. Acesso em Julho de 2024.

NodeJS (2024). Disponível em https://nodejs.org/pt/. Acesso em Julho de 2024.

Ollama (2024). Disponível em https://ollama.com/. Acesso em Julho de 2024.

Oracle (2024). Disponível em https://www.oracle.com/br/cloud/. Acesso em Junho de 2024.

Parsifal (2024). Disponível em https://parsif.al/. Acesso em Maio de 2024.

Quasar Framework (2024). Disponível em https://quasar.dev/. Acesso em Julho de 2024.

RabbitAi (2024). Disponível em https://www.researchrabbit.ai/. Acesso em Maio de 2024.

Validator (2024). Disponível em https://www.npmjs.com/package/validator. Acesso em Julho de 2024.

van de Schoot, R., de Bruin, J., Schram, R., Zahedi, P., de Boer, J., Weijdema, F., Kramer, B., Huijts, M., Hoogerwerf, M., Ferdinands, G., Harkema, A., Willemsen, J., Ma, Y., Fang, Q., Hindriks, S., Tummers, L., and Oberski, D. L. (2021). An open source machine learning framework for efficient and transparent systematic reviews. *Nature Machine Intelligence*, 3(2):125–133.

van Dijk, S. H. B., Brusse-Keizer, M. G. J., Bucsán, C. C., van der Palen, J., Doggen, C. J. M., and Lenferink, A. (2023). Artificial intelligence in systematic reviews: promising when appropriately used. *BMJ Open*, 13(7):e072254.

Vercel (2024). Disponível em https://vercel.com/. Acesso em Junho de 2024.

Vue i18n (2024). Disponível em https://vue-i18n.intlify.dev/. Acesso em Junho de 2024.

Vuefire (2024). Disponível em https://vuefire.vuejs.org/. Acesso em Julho de 2024.

VueJS (2024). Disponível em https://vuejs.org/. Acesso em Julho de 2024.