Estudo Comparativo entre o ChatGPT e o Google no Suporte a Desenvolvedores em Tarefas de Manutenção de Software

Mayara Aiko Teixeira Watanabe

Universidade Federal de Uberlândia
Faculdade de Computação
Bacharelado em Sistemas de Informação

Monte Carmelo - MG
2025

Mayara Aiko Teixeira Watanabe

Estudo Comparativo entre o ChatGPT e o Google no Suporte a Desenvolvedores em Tarefas de Manutenção de Software

Trabalho de Conclusão de Curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Sistemas de Informação.

Área de concentração: Sistemas de Informação

Orientador: Prof. Dr. Adriano Mendonça Rocha

Monte Carmelo - MG
2025

Dedico este trabalho aos meus pais, pelos ensinamentos que me formaram como pessoa. Meus familiares e amigos, que acreditaram em meus sonhos e caminharam ao meu lado para torná-los realidade. E à memória de minha querida avó Misako Watanabe, cuja inspiração e amor seguem vivos em meu coração.

Agradecimentos

Em primeiro lugar, agradeço a Deus, pela força, sabedoria e serenidade concedidas em cada etapa desta jornada.

À minha família, pelo amor incondicional, paciência e incentivo permanente, fundamentais para que eu pudesse seguir em frente mesmo diante dos desafios. Por não me deixarem desistir e por sempre apoiarem meus sonhos, mostrando o quanto eu sou capaz de realizá-los.

Ao professor Dr. Adriano Mendonça Rocha, meu orientador, expresso meus mais sinceros agradecimentos pela orientação precisa, paciência, disponibilidade e apoio durante todo o desenvolvimento deste trabalho. Sua dedicação, competência e incentivo foram essenciais para a construção deste projeto e para o meu crescimento acadêmico e pessoal.

Ao meu namorado, pelo apoio, paciência e pelas palavras de encorajamento nos momentos mais difíceis, especialmente por não permitir que eu desistisse deste sonho. Sua presença constante foi fundamental para que eu superasse os obstáculos e concluísse esta importante etapa da minha vida.

Aos demais professores e colaboradores da instituição, que, ao longo da graduação, compartilharam seus conhecimentos, experiências e valores, contribuindo de forma significativa para minha formação intelectual e ética.

Aos amigos e colegas de curso, pelo companheirismo, pelas trocas de conhecimento e pelos momentos de apoio mútuo que tornaram esta caminhada mais leve e enriquecedora.

"O sucesso nasce do querer, da determinação e persistência em se chegar a um objetivo. Mesmo não atingindo o alvo, quem busca e vence obstáculos, no mínimo, fará coisas admiráveis."

(José de Alencar)

Resumo

Inteligência Artificial tem sido explorada como ferramenta de apoio ao desenvolvimento de sistemas, especialmente na mais longa etapa deste processo: a Manutenção de *Software*. O crescimento do uso do ChatGPT por desenvolvedores motivou sua comparação com métodos tradicionais, como o Google. Este trabalho teve como objetivo avaliar a efetividade do ChatGPT 3.5 e do Google em tarefas de manutenção de *software*. A metodologia envolveu três pesquisadores realizando tarefas nas linguagens de programação Java, Kotlin e Python, sendo que cada linguagem foi atribuída a dois pesquisadores distintos, com ambas as ferramentas, analisando elementos como classes, métodos, variáveis e número de buscas realizadas. As telas dos pesquisadores foram gravadas em vídeo durante a realização de cada tarefa e avaliadas com base em parâmetros objetivos, como inserção, remoção e alteração de elementos do código. Os resultados revelaram que o ChatGPT gerou soluções menos complexas, com cerca de 37,9% mais soluções relevantes e 46,3% menos respostas irrelevantes em comparação ao Google. Além disso, as tarefas realizadas com o ChatGPT apresentaram uma redução média de 30,9% no tempo total de execução, indicando maior objetividade e foco nas respostas fornecidas. Conclui-se que o ChatGPT é capaz de otimizar o fluxo de trabalho, fornecendo respostas diretas e focadas na implementação prática.

Palavras-chave: ChatGPT, Tarefas de Programação, Inteligência Artificial, Manutenção de *Software*, Processamento de Linguagem Natural.

Lista de ilustrações

Lista de tabelas

Lista de siglas

GPT Transformador Pré-treinado Generativo (*Generative Pre-Training Transformer*)

IA Inteligência Artificial (*Artificial Intelligence*)

IDE *Integrated Development Environment* (Ambiente de Desenvolvimento Integrado)

LLM *Large Language Model* (Modelo de Linguagem em Larga Escala)

PLN Processamento de Linguagem Natural (*Natural Language Processing*)

Sumário

Capítulo 1

Introdução

No contexto da programação e das inovações tecnológicas, a Inteligência Artificial (IA) desempenha um papel significativo no auxílio dos desenvolvedores em suas tarefas diárias, seja gerando trechos de código ou ensinando conceitos básicos de programação (ANAGNOSTOPOULOS, 2024). Criado pela organização OpenAI e treinado no supercomputador de IA da Azure (MICROSOFT, 2020), o ChatGPT é um dos modelos mais avançados em Processamento de Linguagem Natural (PLN) e tem se mostrado uma ferramenta eficaz para a resolução de problemas. No entanto, como uma tecnologia emergente, ele levanta diversas problemáticas sobre sua verdadeira eficácia na manutenção de códigos e identificação de erros.

Motivado pela necessidade de melhoria nas ferramentas auxiliares na programação, este trabalho tem como objetivo investigar o uso do ChatGPT 3.5 para auxiliar desenvolvedores em pares de tarefas de programação. Estes pares consistem em realizar a mesma tarefa de duas formas diferentes, uma vez executando-a com o Google como ferramenta auxiliar e outra tendo o ChatGPT como auxiliador, a fim de comparar a complexidade dos dois resultados. Serão aplicadas técnicas de avaliação em tarefas de manutenção de código para compreender como pode-se aplicar a IA no contexto da programação. Por meio deste estudo, espera-se contribuir para o avanço do conhecimento sobre o uso do ChatGPT e de IAs na manutenção e criação de códigos e explorar o potencial dessa tecnologia nos dias atuais, contribuindo para o avanço, eficiência e qualidade do desenvolvimento de *software*.

1.1 Motivação

Ao trabalhar no desenvolvimento e na manutenção de *softwares*, os programadores frequentemente se deparam com erros desconhecidos e precisam de uma fonte confiável para encontrar a melhor solução possível.

Recorrer a livros de programação muitas vezes não possibilita a solução do problema propriamente dito, já que geralmente aborda apenas a sintaxe básica e aplicações sim-

ples (GOMES; HENRIQUES; MENDES, 2008), raramente fornecendo instruções para soluções de possíveis problemas.

Assim, plataformas *online* de perguntas e respostas como o Stack Overflow[1] começaram a ser utilizadas como forma de solucionar esses problemas mais práticos que os materiais didáticos não abordavam, permitindo que programadores iniciantes pudessem aprender com usuários da plataforma mais experientes.

Em casos de erros não recorrentes, os usuários que necessitam de respostas mais imediatas para o problema e não as encontram em sites como o Stack Overflow, recorrem a motores de busca, como o Google. A vantagem é que os buscadores retornam uma grande lista de possíveis soluções, porém já foi estudado que as melhores soluções nem sempre se encontram nas primeiras páginas (ROCHA; MAIA, 2023), o que pode consumir muito tempo de desenvolvimento em problemas relativamente simples.

Com o avanço da tecnologia e o surgimento das IAs, muitos programadores enxergaram os *ChatBots*, tal como o ChatGPT, como oportunidade de resolução de erros. Entretanto, como toda tecnologia emergente, é necessário entender melhor os usos e as limitações do ChatGPT para atingir o maior nível de eficiência na construção e manutenção de códigos.

Este trabalho tem como motivação trazer esclarecimentos sobre tais usos e limitações e comparar as diferentes técnicas de busca de informações para a manutenção de código, enfatizando a diferença entre o Google e o ChatGPT 3.5.

1.2 Problema

No desenvolvimento e manutenção de *software*, desenvolvedores enfrentam frequentemente desafios que vão além do conhecimento básico. A identificação e correção de erros, bem como a otimização de código, demandam tempo e recursos. Tradicionalmente, soluções para esses problemas são buscadas em livros, fóruns *online*, ou por meio de motores de busca. Contudo, essa abordagem apresenta limitações, como a necessidade de navegar por diversas páginas e validar informações fragmentadas, o que pode aumentar o tempo de resolução.

Com o surgimento de ferramentas de inteligência artificial, como o ChatGPT, os desenvolvedores passaram a ter uma alternativa para obter respostas mais rápidas e focadas. No entanto, apesar do potencial dessas tecnologias, sua efetividade ainda não é totalmente compreendida. Em particular, não está claro se o ChatGPT é capaz de superar as limitações dos motores de busca tradicionais na manutenção de código e se as respostas geradas são suficientemente precisas e completas para evitar retrabalho.

Portanto, o problema central deste estudo reside em investigar as vantagens e limitações do ChatGPT como ferramenta de apoio na programação, em comparação com os motores de busca. É essencial compreender se o uso do ChatGPT pode otimizar o

[1] Disponível em: <https://stackoverflow.com>. Acesso em: 14 fev. 2024.

processo de desenvolvimento ao reduzir o tempo necessário para encontrar soluções e se a reutilização dos códigos gerados pela IA é prática e eficaz. Além disso, este trabalho busca avaliar se há um impacto significativo na complexidade do código gerado ao utilizar diferentes fontes de apoio, determinando qual ferramenta oferece melhores resultados na prática.

1.3 Hipótese

Este trabalho apresenta as seguintes hipóteses:

- ❏ O ChatGPT fornece soluções mais eficientes para tarefas de manutenção de *software* quando comparado com motores de busca.
- ❏ Existe um padrão de comportamento dos desenvolvedores que utilizam a ferramenta ChatGPT para auxiliar na programação.

1.4 Objetivos

O **objetivo geral** deste trabalho é mensurar a capacidade de modelos de IAs generativas pré-treinadas usadas como ferramenta de auxílio para a manutenção de *softwares* e comparar a complexidade da reutilização dos códigos gerados pelo ChatGPT e pelo Google.

Para isso, os seguintes **objetivos específicos** foram traçados:

- ❏ Analisar a maneira que os desenvolvedores interagem com o ChatGPT ao realizar a busca por solução de tarefas voltadas à manutenção de *software*.
- ❏ Identificar a diferença entre o uso de IAs generativas e motores de busca na realização de tarefas de manutenção de *software*.
- ❏ Comparar o tempo que os desenvolvedores levam para encontrar soluções em problemas de manutenção de *software* usando o ChatGPT e o Google.
- ❏ Analisar o número de perguntas necessárias para que o ChatGPT forneça o resultado desejado para problemas de manutenção de *software*.
- ❏ Investigar a prática dos desenvolvedores na reutilização de códigos gerados pelo ChatGPT.

1.5 Organização da Monografia

A monografia está organizada da seguinte maneira:

- Capítulo 1 – Introdução

  A Introdução é a seção inicial do trabalho, onde é apresentado o tema da monografia, justificando sua relevância, contextualizando o problema de pesquisa, descrevendo a motivação para realizar o estudo e apresentando as hipóteses ou perguntas de pesquisa que guiarão o desenvolvimento do trabalho. Além disso, são expostos os objetivos gerais e específicos da pesquisa, que orientam a metodologia e as análises a serem realizadas.

- Capítulo 2 – Fundamentação Teórica

  O capítulo de Fundamentação Teórica apresenta os conceitos e teorias fundamentais, como "Manutenção de *Software*", "Motores de Busca"e "Arquitetura *Transformer*", que sustentam a pesquisa, proporcionando o embasamento necessário para compreensão do problema. Também são apresentados estudos anteriores que são relevantes para o trabalho, ajudando a situar a pesquisa no cenário acadêmico.

- Capítulo 3 – Método e Análise dos Resultados

  O capítulo é dividido em duas seções: o Método descreve como a pesquisa foi conduzida, desde a coleta de dados, as ferramentas e algoritmos utilizados e como estes dados foram tratados e analisados. Já a Análise dos Resultados apresenta os resultados obtidos a partir da aplicação do método, interpreta os resultados e discute suas implicações dentro do contexto da pesquisa.

- Capítulo 4 – Conclusão

  Este capítulo tem um papel muito importante de sintetizar e refletir sobre as contribuições do trabalho. Ele não só recapitula as principais descobertas e contribuições, mas também aponta as direções para futuras pesquisas e possíveis melhorias no campo estudado.

- Referências Bibliográficas

  Um conjunto de fontes consultadas durante a realização da pesquisa, com base no conhecimento já existente na área.

Capítulo 2

Fundamentação Teórica

Neste capítulo, serão apresentados os conceitos e teorias fundamentais que embasam o trabalho proposto, bem como os trabalhos correlatos que se assemelham e diferenciam da solução a ser apresentada.

2.1 Conceitos Básicos

Nesta seção, serão explorados conceitos fundamentais que sustentam o desenvolvimento do trabalho, incluindo a Manutenção de *Software*, Reuso de *Software*, Motores de Busca abordando o algoritmo PageRank, Processamento de Linguagem Natural (PLN), Inteligência Artificial (IA), Modelos de Linguagem de Grande Escala (LLM), Arquitetura *Transformer*, e o modelo GPT. Esses tópicos são essenciais para entender a base teórica que orienta a solução proposta, além de destacar as inovações e desafios relacionados a cada área.

2.1.1 Manutenção de *Software*

A Manutenção de *Software* é uma etapa da Engenharia de *Software* que consiste em modificar, aprimorar ou aperfeiçoar parte do *software* após sua entrega. É fundamental para que, com o passar do tempo, sistemas possam acompanhar novas tendências tecnológicas. A Manutenção de *Software* é muitas vezes a etapa de maior custo durante o desenvolvimento e, apesar de ser considerada de grande importância por desenvolvedores, ainda é necessária uma maior atenção para melhorar a sua eficiência, visto que atualmente as metodologias ágeis vêm tomando conta do mercado de tecnologia (SOUZA; ANQUETIL; OLIVEIRA, 2006). São caracterizados três tipos de atividades de manutenção, sendo elas:

- Manutenção corretiva (realizada em resposta à identificação de falhas);

- ❏ Manutenção adaptativa (realizada antecipando mudanças nos ambientes de dados ou de processamento);
- ❏ Manutenção perfectiva (realizada para eliminar ineficiências, melhorar o desempenho ou aumentar a facilidade de manutenção).

Por isso, de acordo com a pesquisa de Lientz, Swanson e Tompkins (1978), a manutenção e a melhoria dos sistemas consomem uma parte significativa dos recursos de um projeto, reforçando a ideia de que a maior parte do ciclo de vida de um *software* está na manutenção, não no desenvolvimento inicial, mas reforçando que o investimento em qualidade de *software* e boa documentação são de grande importância para tornar a manutenção mais eficiente.

2.1.2 Reuso de *Software*

O Reuso de *Software* é o processo de criar um *software* baseando-se em componentes, ideias ou processos já existentes e aplicando as adaptações necessárias, tendo como principal motivação reduzir o tempo e esforço ao criar *softwares* de larga escala, fornecendo soluções às subtarefas (KRUEGER, 1992).

O reuso de *software* pode ser abordado em diferentes níveis, dependendo do tipo de artefato reutilizado. As principais abordagens incluem o reuso de código-fonte, seja por trecho de código ou bibliotecas, reuso de componentes, onde módulos são desenvolvidos para serem reaproveitados em mais projetos e arquiteturas de *software*, que são padrões de estrutura utilizados para organizar e estruturar um projeto de grande escala.

Uma das maiores dificuldades encontradas ao reutilizar trechos e arquiteturas de outros projetos é a integração e compatibilidade de diferentes estilos de programação, padrões e tecnologias, que podem conflitar entre si ou com o sistema base. Com os avanços da IA, novas abordagens têm sido exploradas para otimizar o reuso de *software*. Algoritmos de aprendizado de máquina e PLN podem facilitar a busca, adaptação e integração de componentes reutilizáveis, reduzindo a barreira da distância cognitiva e tornando o reuso mais eficiente e acessível.

2.1.3 Motores de Busca

Um motor de busca é um programa projetado para pesquisar em sua própria base de dados resultados baseados em palavras-chave fornecidas pelos usuários, permitindo a recuperação de informações em grande escala. Plataformas como Google, que oferecem esse mecanismo de busca, normalmente possuem algoritmos de ranqueamento de páginas de acordo com as palavras-chave buscadas pelo usuário e retornam páginas *web*, documentos textuais, vídeos, imagens, entre outros formatos de dados (SEYMOUR; FRANTSVOG; KUMAR, 2011). Seu principal objetivo é recuperar conteúdos relevantes a partir de

vastos bancos de dados indexados, facilitando a navegação e o acesso à informação. O funcionamento de um motor de busca pode ser dividido em três etapas principais:

O processo de rastreamento (*Web Crawling*) é conduzido por programas automatizados chamados *web crawlers*, *spiders* ou *bots*. Esses programas navegam pela internet coletando dados de páginas da *web*, seguindo *links* e descobrindo novos conteúdos. O rastreamento pode ser orientado por arquivos que informam aos motores de busca quais páginas devem ser visitadas. Grandes motores de busca, como Google, Bing e Yahoo, possuem seus próprios rastreadores, como o *Googlebot*.

Após o rastreamento, as páginas visitadas passam por um processo de indexação, onde seu conteúdo é analisado e armazenado em um banco de dados estruturado. Durante a indexação, os motores de busca extraem palavras-chave de títulos, subtítulos, metadados e do corpo do texto. Alguns motores de busca armazenam não apenas palavras-chave, mas também cópias completas das páginas, permitindo que os usuários visualizem versões arquivadas em cache. A indexação também pode considerar fatores como a estrutura do HTML, tempo de carregamento da página e uso de imagens e vídeos.

Quando um usuário insere uma consulta no motor de busca, este recupera as páginas mais relevantes da sua base de dados e as exibe em uma ordem de relevância. Para isso, os motores de busca realizam o processo de classificação, feito por meio de algoritmos complexos que consideram fatores como relevância do conteúdo em relação à consulta realizada, autoridade da página, medida pela quantidade e qualidade dos *links* que apontam para ela, experiência do usuário, incluindo métricas como taxa de rejeição e tempo de permanência na página.

2.1.3.1 PageRank

O Google utiliza um algoritmo de ranqueamento chamado PageRank, desenvolvido por Larry Page e Sergey Brin para classificar páginas da web com base em sua importância relativa dentro da estrutura de *links* da internet. Ele foi um dos principais fatores que contribuíram para o sucesso do Google como motor de busca, diferenciando-o dos concorrentes que utilizavam apenas a correspondência textual para ranquear páginas. O PageRank atribui um *score* de relevância a cada página da web com base na estrutura de *links* entre elas. O algoritmo se baseia em um modelo de navegação aleatória, onde um usuário fictício clica nos *links* de uma página de maneira probabilística. O funcionamento pode ser dividido em três principais componentes matemáticos:

Observando a Figura 1 do artigo de Wills (2006), a web é representada como um grafo direcionado, onde os nós representam páginas da web e as arestas representam *links* entre páginas. Se uma página i contém um *link* para uma página j, então existe uma aresta direcionada de i para j.

Wills (2006) ainda explica que a matriz de transição H define a probabilidade de um

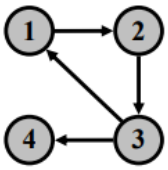

Figura 1 – Grafo direcionado com quatro nós.
Fonte: Wills (2006)

usuário navegar de uma página para outra:

$$H_{ij} = \begin{cases} \frac{1}{L_i}, & \text{se a página } i \text{ tem um } \textit{link} \text{ para } j \\ 0, & \text{caso contrário} \end{cases} \tag{1}$$

onde L_i é o número total de *links* de saída da página i. Para resolver o problema dos nós sem *links* de saída (*dangling nodes*), substituímos a linha correspondente por um vetor de distribuição uniforme w:

$$S = H + dw \tag{2}$$

onde d indica os nós pendentes. A matriz **Google Matrix** G modela o comportamento de um usuário real:

$$G = \alpha S + (1 - \alpha)1v \tag{3}$$

com $\alpha = 0.85$ sendo o *damping factor*, que representa a probabilidade do usuário seguir um *link*.

O vetor PageRank π é obtido resolvendo:

$$\pi G = \pi \tag{4}$$

ou seja, π é o **autovetor dominante** da matriz G.

2.1.4 Processamento de Linguagem Natural (PLN)

O PLN é uma série de técnicas computadorizadas para analisar, representar e compreender textos em linguagem humana. Essa análise ocorre em níveis sintáticos, semânticos e pragmáticos, por isso é capaz de alcançar uma capacidade de processamento semelhante à humana. A área de pesquisa de PLN envolve técnicas como aprendizado de máquina e processamento de dados, para que o computador saiba lidar com a complexidade e a variedade linguística, comunicando-se melhor com o usuário e de maneira personalizada (LIDDY, 2001).

2.1.5 Inteligência Artificial (IA)

O termo IA, que foi inicialmente cunhado para a teoria de que a inteligência humana poderia ser exibida por máquinas, atualmente tem uma definição muito mais aplicável no

cotidiano como uma avaliadora de conjuntos de dados extremamente grandes em frações mínimas de tempo, tornando-se facilitadora em diversos aspectos da sociedade, como recomendações personalizadas de anúncios, carros autônomos, entre outras tecnologias análogas a IA (HELM et al., 2020).

Por ter um conceito muito vasto, o termo IA é, neste Trabalho de Conclusão de Curso, utilizado especificamente no contexto de IA generativa, com foco em Modelos de Linguagem de Grande Escala (LLMs). Assim, todas as referências à IA ao longo deste trabalho estão alinhadas com essa definição operacional, que considera os LLMs como sistemas capazes de analisar e gerar código, responder perguntas e realizar outras tarefas linguísticas de forma autônoma.

2.1.6 *Large Language Model* (LLM)

Os Modelos de Linguagem de Grande Escala (LLM), segundo a pesquisa de Shanahan (2024), são sistemas de inteligência artificial que possuem capacidade de analisar e gerar código de forma autônoma, utilizando um modelo matemático generativo que representa a distribuição estatística de sequências de *tokens* em um vasto corpo de texto gerado por humanos, podendo incluir palavras, partes de palavras, caracteres individuais e pontuação. O objetivo principal dos LLMs é prever o próximo *token* com base no texto fornecido, o que permite aplicações como agentes conversacionais, tradutores automáticos e sistemas de geração de texto.

Dados relevantes da mesma pesquisa mostram que os LLMs demonstram um desempenho que melhora conforme o conjunto de dados de treinamento cresce. À medida que os modelos escalam, ocorrem avanços qualitativos inesperados, possibilitando a realização de tarefas mais complexas, simulando um raciocínio humano. Contudo, o artigo alerta para o perigo de descrever LLMs em termos humanos, como “acreditar” ou “saber”, pois isso pode levar à superestimação de suas capacidades. Um LLM não possui crenças, intenções ou compreensão; ele apenas gera sequências de palavras estatisticamente prováveis.

Apesar de se basearem apenas na previsão do próximo *token*, os LLMs conseguem realizar diversas tarefas, desde responder perguntas até criar textos coerentes, devido à natureza abrangente e diversificada dos dados de treinamento. Quando integrados a sistemas que acessam fontes externas de informação, como bancos de dados factuais, ou que possuem interação com o ambiente, como robôs, eles podem adquirir um comportamento que se assemelha mais ao de um agente com conhecimento.

2.1.7 Arquitetura *Transformer*

A arquitetura *Transformer* é um modelo baseado em redes neurais que revolucionou o Processamento de Linguagem Natural (PLN) por meio do mecanismo de autoatenção, permitindo que cada elemento da sequência processe informações globais de maneira efi-

ciente e paralelizável. Como diferencial em comparação com outras arquiteturas, a Arquitetura *Transformer* representa uma inovação paradigmática na modelagem de sequências, eliminando a necessidade de redes recorrentes (RNNs) e convolucionais (CNNs).

Graças a essa inovação, as técnicas de PLN avançaram significativamente, permitindo que modelos baseados na arquitetura *Transformer* interpretassem a importância contextual de cada elemento da entrada solicitada pelo usuário e gerassem respostas mais coerentes e contextualmente relevantes. Esse avanço possibilitou uma compreensão mais profunda de textos complexos, superando abordagens que se limitavam à correspondência de palavras-chave (HAROON; C.A.; A.S., 2023).

Além do sucesso na área de PLN, recentes pesquisas expandiram a aplicação dos *Transformers* para domínios como visão computacional, séries temporais, processamento de áudio e até mesmo grafos. Os experimentos realizados no estudo de Min et al. (2022) demonstraram que essas modificações não apenas preservam as vantagens da autoatenção global do *Transformer*, mas também aumentam sua capacidade de aprendizado sobre relações topológicas complexas em grafos, resultando em um desempenho superior em diversas tarefas que exigem um grande volume de dados.

A arquitetura do *Transformer* consiste em dois blocos fundamentais: Codificador e Decodificador, que funcionam de maneira hierárquica para aumentar a capacidade do modelo em comparação com os já pré-existentes.

A adoção dessa arquitetura trouxe avanços substanciais para a tradução automática e, posteriormente, foi amplamente aplicada a outras áreas, incluindo visão computacional, bioinformática e modelagem de séries temporais. A Figura 2 do artigo de Vaswani et al. (2023) ilustra a arquitetura completa do *Transformer*, evidenciando seus componentes principais e suas interconexões. O lado esquerdo da figura representa o codificador, responsável por processar a entrada e gerar representações latentes. Já o direito representa o decodificador, que recebe os *embeddings* processados pelo codificador e gera a sequência de saída.

Cada camada do **codificador** é composta por:

- ❏ Mecanismo de Autoatenção Multi-Cabeça (*Multi-Head Self-Attention*): Permite que a entrada seja processada considerando múltiplos contextos simultaneamente.
- ❏ Camada de *Feed-Forward* (FFN): Aplicada separadamente a cada posição da sequência, refinando a representação aprendida.
- ❏ Camada de Normalização e Conexões Residuais: Asseguram estabilidade na propagação do gradiente e evitam o desaparecimento de informações críticas.

A entrada inicial passa por uma camada de *embedding*, que converte *tokens* em vetores de dimensão fixa. Adicionalmente, são aplicadas codificações posicionais para garantir que a ordem da sequência seja preservada.

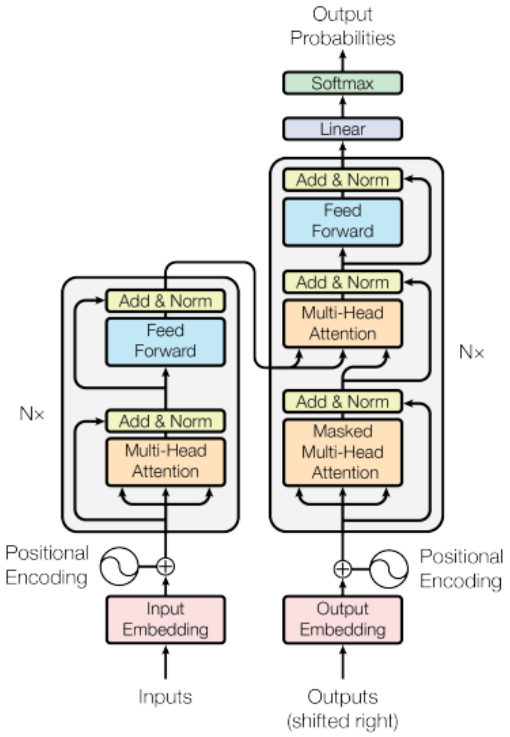

Figura 2 – Arquitetura completa do *Transformer*
Fonte: Vaswani et al. (2023)

Além dos componentes já presentes no codificador, o decodificador contém um módulo adicional de atenção cruzada (*Encoder-Decoder Attention*), permitindo que cada posição do decodificador acesse informações codificadas da entrada.

Outro elemento fundamental do decodificador é a máscara causal no mecanismo de autoatenção. Essa máscara impede que posições futuras sejam acessadas durante a predição, garantindo a natureza autoregressiva do modelo.

O fluxo geral de processamento dos dados, de acordo com a Figura 2 segue os seguintes passos:

1. A entrada passa por uma camada de *embedding*, seguida da adição de codificações posicionais.

2. O codificador processa os *embeddings* por meio de múltiplas camadas de autoatenção e redes *feed-forward*, gerando uma representação intermediária.

3. O decodificador recebe como entrada os embeddings da saída (*shifted right*) e, por meio do mecanismo de atenção cruzada, acessa informações do codificador.

4. A sequência gerada passa por camadas adicionais de normalização e *feed-forward* até a obtenção da previsão final.

2.1.8 *Generative Pre-training Transformer* (GPT)

O modelo Transformador Pré-treinado Generativo (GPT) utiliza apenas da parte decodificadora da arquitetura *transformer*. Como um exemplo, podemos utilizar o GPT-3, um modelo de linguagem autoregressivo, treinado com uma larga escala de parâmetros para produzir texto semelhante ao humano (TOPAL; BAS; HEERDEN, 2021). O GPT-3 é um modelo de inteligência artificial desenvolvido pela OpenAI que representa um avanço significativo na compreensão e geração de linguagem natural. Com um total de 175 bilhões de parâmetros, é capaz de realizar tarefas complexas de processamento de texto com uma precisão impressionante. Seu desenvolvimento baseia-se na arquitetura *Transformer*, utilizando aprendizado profundo para analisar padrões na linguagem humana e gerar respostas coerentes e contextualizadas, como comenta Zhang e Li (2021).

A principal abordagem do GPT-3 é o aprendizado de poucos exemplos (*few-shot learning*), que permite que ele execute diversas tarefas com um mínimo de instruções ou exemplos prévios. Em alguns casos, ele é capaz de operar em um contexto de *zero-shot learning*, ou seja, sem qualquer treinamento específico para a tarefa desejada. Isso o torna altamente flexível para aplicações diversas, como resposta a perguntas, geração de textos, resumo de informações, análise semântica e até mesmo geração de códigos de programação, fazendo com que o GPT-3 fosse considerado uma das 10 tecnologias mais revolucionárias de 2021 pelo *MIT Technology Review* devido à sua capacidade de transformar a interação entre humanos e máquinas.

O GPT-3 utiliza um método de pré-treinamento seguido de ajuste fino (*fine-tuning*) para aprender e produzir texto de maneira autônoma. Seu treinamento foi realizado com uma vasta quantidade de dados provenientes da internet, incluindo livros, artigos e outros documentos, permitindo que o modelo aprenda padrões gramaticais, estilísticos e semânticos da linguagem natural.

2.2 Trabalhos Relacionados

Nota-se um aumento nos trabalhos atuais que abordam o ChatGPT como ferramenta facilitadora de desenvolvimento e manutenção de *software*. Dentre eles, trabalhos que focam em mostrar desafios e oportunidades do uso de tecnologias generativas no desenvolvimento de *software*, como no trabalho de Rahmaniar (2023), que explica vantagens e limitações percebidas nas principais linguagens de programação, assim como comparações de desempenho entre IAs generativas e aponta os maiores defeitos e qualidades do ChatGPT. Os autores explicam que, para realizar a manutenção de código utilizando IA, o tempo para tal etapa na Engenharia de *Software* seria reduzido drasticamente; entretanto, com o passar do tempo, sua complexidade aumentaria, podendo gerar dilemas de versionamento. O estudo destaca a facilidade de compreensão do usuário ao utilizar a ferramenta, graças à sua capacidade de gerar textos mais humanizados. Este trabalho

abordará esse aspecto, comparando a eficácia da ferramenta com o uso do Google como mecanismo de busca.

Já na pesquisa de Dantas, Rocha e Maia (2023), é mostrado o uso do ChatGPT em geração de trechos de código (*Code Snippets*), fazendo um estudo comparativo com trechos gerados pela IA e os fornecidos pela plataforma CROKAGE, que utiliza como base de dados o *Stack Overflow*. O estudo foi realizado na linguagem Java e notou-se alta efetividade na geração de códigos didáticos em comparação com códigos feitos por humanos. Usando a ferramenta *SonarQube*, um *software* de análise estática, foram apontados inúmeros erros de ambas as plataformas, concluindo que, apesar da comparação favorável ao ChatGPT, ainda há necessidade de outras análises e melhorias.

Ebert e Louridas (2023) fazem o estudo de IAs Generativas para aumentar a produtividade na Engenharia de *Software*. O estudo apontou, dentre outras informações relevantes, que tecnologias generativas existem há muitos anos, mas a adoção em massa é recente. O ChatGPT levou apenas dois meses para atingir a marca de 100 milhões de usuários e tem potencial de transformar a maneira de desenvolver *softwares*, mas também apresenta desafios de segurança e confiabilidade.

Como diferencial, este trabalho apresentará uma análise comparativa entre o uso do ChatGPT e de Motores de Busca. Como foi mostrado em Rocha e Maia (2023), os primeiros resultados de pesquisas voltadas a tarefas de programação em Motores de Busca nem sempre são os mais relevantes, sendo o Google a *engine* de foco da pesquisa, o trabalho mostra que das 20 principais páginas retornadas para 10 consultas diferentes, apenas 31% eram relevantes para desenvolvedores. Os autores propõem abordagens de mineração para remover as páginas irrelevantes, destacando a importância de um filtro de pesquisa personalizado para conteúdo de desenvolvimento.

No método deste trabalho, serão aplicadas técnicas semelhantes às apresentadas no terceiro capítulo da tese de Rocha (2022), que aborda a análise preliminar das 20 principais páginas retornadas pelo mecanismo de pesquisa do Google para 14 *queries* distintas. Foram observadas 16 amostras de tarefas em Java, divididas em pares semelhantes, para diferenciar *rankings* de alta e baixa qualidade. Como resultado, diversas páginas foram consideradas irrelevantes para o problema, destacando que:

- ❏ Para a classificação de alta qualidade, as páginas relevantes estão no topo da classificação, economizando tempo para os desenvolvedores, pois encontram soluções rapidamente, sem precisar analisar páginas irrelevantes.

- ❏ Para a classificação de baixa qualidade, os desenvolvedores gastam mais tempo analisando as páginas, à medida que as soluções relevantes estão espalhadas, necessitando visitar mais páginas para encontrar a solução desejada.

De forma semelhante, neste trabalho, serão analisados quantitativamente parâmetros como o número de classes, métodos e variáveis inseridas, removidas ou alteradas, com a

intenção de comparar a complexidade de utilização do ChatGPT e do Google como ferramentas auxiliares. Para isso, serão utilizados pares de tarefas funcionalmente equivalentes, atribuídos a diferentes participantes, nas quais uma das tarefas é realizada com o apoio do ChatGPT e a outra com o Google, permitindo uma análise comparativa controlada entre as ferramentas.

Outro trabalho relevante é o de Russo (2023), um estudo de caso para extrair o interesse de desenvolvedores no uso de *Large Language Model* (LLM) no contexto da Engenharia de *Software*. O estudo apresentou um questionário analisando qualitativamente a intenção, a facilidade de uso e a utilidade percebida. Como resultado, grande parte dos engenheiros de *software* demonstra interesse em adotar LLMs para melhoria e manutenção de código, bem como para eficiência e automação de tarefas repetitivas, implementação de padrões de *design* e princípios SOLID, além de resolução de problemas, como compreensão de códigos confusos ou mal documentados. O estudo permitiu compreender o comportamento do desenvolvedor diante do uso do ChatGPT na Engenharia de *Software*, o que será abordado de forma prática neste estudo.

No contexto de manutenção de *software* temos o trabalho de Nunes et al. (2025), que investiga a eficácia dos LLMs em resolver problemas de manutenibilidade de código em projetos reais. Semelhante a este trabalho, a pesquisa analisa como esses modelos identificam e corrigem questões que afetam a legibilidade, modularidade e simplicidade do *software*, fatores críticos para garantir a longevidade e a qualidade do código. Para isso, os autores aplicaram LLMs a repositórios reais de *software*, avaliando o impacto das mudanças sugeridas em métricas como complexidade ciclomática e conformidade com boas práticas de codificação. Os resultados mostraram que os modelos conseguem oferecer melhorias relevantes, mas frequentemente sugerem alterações excessivas ou inadequadas, exigindo validação humana. A principal conclusão do artigo é que, embora os LLMs possam ser aliados úteis na manutenção de *software*, ainda há desafios a serem superados, como a falta de contexto sobre a arquitetura do sistema.

O artigo de Rajbhoj et al. (2024) analisa como o ChatGPT pode acelerar o desenvolvimento de *software*, explorando sua aplicabilidade em diferentes fases do processo, desde a escrita de código até a depuração e documentação. A pesquisa se baseia em um estudo de caso onde desenvolvedores utilizam o ChatGPT para automatizar tarefas repetitivas, gerar trechos de código e melhorar a produtividade. A metodologia envolveu a observação do desempenho de programadores com e sem o uso da IA, medindo a velocidade de entrega e a qualidade do código produzido. Os resultados indicam que o ChatGPT pode, de fato, reduzir o tempo necessário para determinadas atividades, mas ainda apresenta limitações, como a propensão a gerar código incorreto ou subótimo. A principal conclusão do estudo é que a ferramenta pode ser um recurso valioso para desenvolvedores, mas não substitui a revisão humana e a compreensão profunda do código. Comparado a um estudo voltado exclusivamente para manutenção de *software*, este artigo tem um foco mais

amplo, abordando todas as etapas do desenvolvimento.

De maneira semelhante ao proposto neste trabalho, o estudo de Xu, Feng e Chen (2023) investiga comparativamente a eficácia do ChatGPT e do Google na recuperação de informações, considerando critérios como precisão, relevância, experiência do usuário e confiabilidade das respostas. A pesquisa conduz experimentos nos quais diferentes perguntas foram submetidas a ambas as plataformas, cujas respostas foram analisadas quanto à completude, clareza e rigor informacional. Os resultados demonstram que o ChatGPT oferece uma experiência mais fluida e respostas estruturadas de maneira direta, embora sua confiabilidade seja inferior à do Google, que, por sua vez, permite a verificação de fontes e apresenta informações com maior precisão, especialmente em tópicos factuais e atualizados. O estudo concluiu que o ChatGPT pode ser utilizado como um assistente inicial na organização de informações, mas não substitui integralmente os mecanismos de busca tradicionais. Este trabalho busca resultados voltados à área de Manutenção de *Software* utilizando um comparativo similar.

O artigo de Jin et al. (2024) propõe e descreve o delineamento de um estudo empírico com o objetivo de avaliar a utilidade do ChatGPT como ferramenta de apoio ao desenvolvimento de *software*, especialmente na geração de código, destacando a necessidade de investigações sistemáticas para compreender seu impacto prático. O método consiste em um experimento com estudantes de ciência da computação que realizarão tarefas de programação utilizando ou não o ChatGPT, com o desempenho sendo avaliado em termos de tempo de execução, corretude do código e percepção subjetiva dos participantes quanto à utilidade da ferramenta. O artigo apresenta um conjunto de tarefas planejadas, uma análise do desenho experimental, critérios de avaliação e ferramentas utilizadas para coleta e análise dos dados. Contudo, o estudo ainda está em fase de planejamento e não apresenta resultados empíricos concretos.

Outro artigo relevante no contexto do uso do ChatGPT é o de Moratis et al. (2024) que investiga a eficiência e a qualidade do código gerado pelo ChatGPT em contextos de desenvolvimento de *software*, com base no conjunto de dados DevGPT (XIAO et al., 2024), que reúne conversas entre desenvolvedores e o modelo, além dos *commits* resultantes dessas interações. Inicialmente, os autores realizaram uma filtragem e pré-processamento dos dados, identificando o *JavaScript* como a linguagem predominante, e categorizaram as conversas em cinco tipos, focando nas categorias “Escreva este código” e “Melhore este código”. A análise revelou que, embora o ChatGPT consiga gerar código com relativa facilidade, cerca de 25% dos blocos continham violações, sendo a maioria relacionada a estilo ou boas práticas e apenas 11,9% classificadas como potencialmente prejudiciais. Além disso, ao comparar o código anterior e posterior à intervenção do ChatGPT, observou-se uma tendência de redução nas violações, indicando que o modelo pode, em alguns casos, melhorar a qualidade do código existente. Contudo, os autores reconhecem limitações, como o viés otimista do conjunto de dados, que exclui interações mal-sucedidas. O es-

tudo conclui que, embora promissor, o uso do ChatGPT deve ser criterioso, com validação humana para garantir a integridade e qualidade do código gerado.

Por fim, também é possível mencionar o artigo de Shuvo et al. (2025), que realiza uma análise empírica aprofundada sobre as capacidades de geração de código do ChatGPT, comparando seu desempenho em problemas de programação com contextos curtos e contextos longos e narrativos. Utilizando um conjunto de 102 problemas do *LeetCode* (contextos curtos) e 150 do *Codeforces* (contextos longos), distribuídos entre níveis de dificuldade (fácil, médio e difícil) e datas de publicação (antes e depois de setembro de 2021). Os autores avaliam a taxa de acertos, erros e a eficiência do código gerado em *Python* e *C++*. Os resultados mostram um desempenho significativamente superior em problemas concisos, sendo que a complexidade narrativa dos problemas do *Codeforces* gerou maiores dificuldades de compreensão e extração de informações relevantes pelo modelo. O estudo também inclui uma segunda rodada de testes com reengenharia de *prompts* e inserção de *feedback* baseado em erros anteriores, o que melhorou ligeiramente a performance em *LeetCode*, mas reduziu a performance em *Codeforces*, que já era mais baixa. A análise da eficiência computacional revelou que *prompts* refinados contribuíram para melhorar o tempo de execução, especialmente em *C++*. Os autores concluem que o ChatGPT enfrenta sérios desafios em contextos mais complexos e exige abordagens refinadas para alcançar resultados confiáveis e otimizados, contudo ao compreender a forma de se estruturar o *prompt*, a ferramenta se mostra eficiente e capaz de solucionar problemas até um certo nível de complexidade.

Capítulo 3

Método e Análise dos Resultados

Este capítulo aborda o método e os resultados deste estudo, detalhando os parâmetros de avaliação utilizados, assim como os dados relevantes obtidos.

3.1 Método para a Avaliação

O presente trabalho propõe a avaliação dos resultados de um estudo conduzido por um grupo composto por quatro pesquisadores, onde três deles realizaram tarefas de manutenção de código em três diferentes linguagens de programação: Python, Java e Kotlin. A linguagem Java foi escolhida por sua consolidação tanto no setor industrial quanto no meio acadêmico, além de possuir uma ampla base de conhecimento técnico e vasta documentação disponível. A linguagem Python foi selecionada por representar uma abordagem moderna e amplamente adotada na atualidade, destacando-se por sua simplicidade e grande popularidade entre desenvolvedores. Por fim, a linguagem Kotlin foi incluída por sua forte presença no desenvolvimento de aplicações móveis, especialmente para a plataforma *Android*, permitindo a avaliação das ferramentas também nesse contexto específico de desenvolvimento.

A pesquisa foi dividida em três fases principais, conforme mostra a Figura 3. Durante a

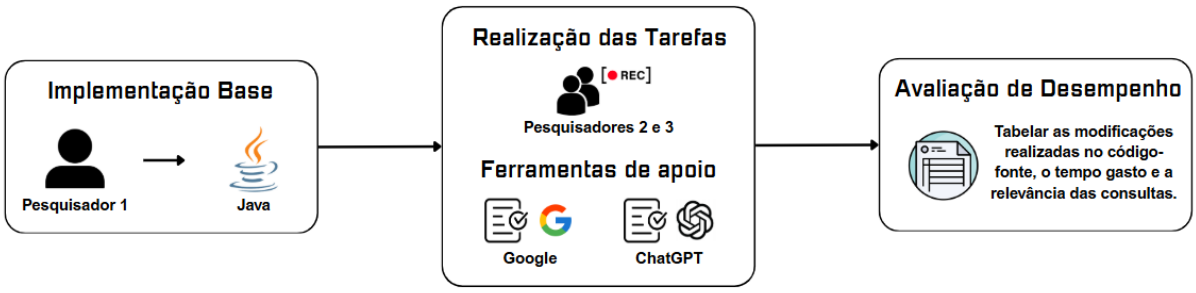

Figura 3 – Visão geral das fases da pesquisa.
Fonte: Autoria própria.

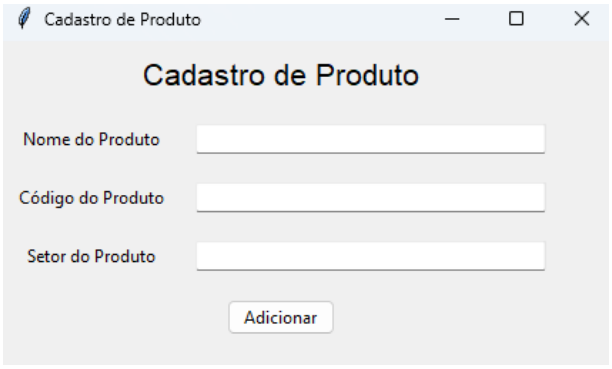

Figura 4 – Tela da implementação base da Tarefa 1.
Fonte: Autoria própria.

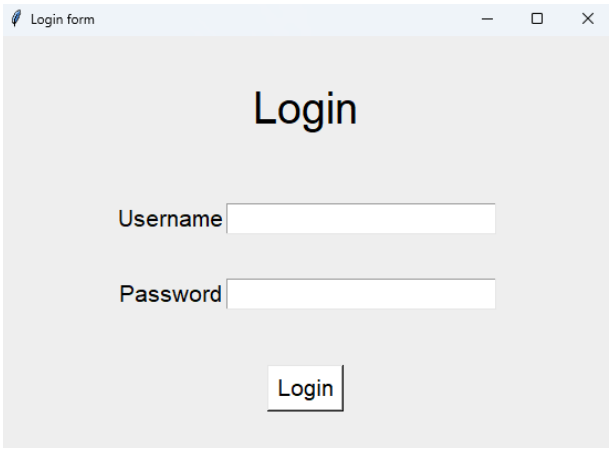

Figura 5 – Tela da implementação base da Tarefa 2.
Fonte: Autoria própria.

fase inicial da pesquisa, cada participante foi incumbido de criar a implementação base de duas tarefas padronizadas, replicadas nas três linguagens mencionadas. A primeira tarefa será voltada a um modelo simples de cadastro de produtos de mercado, como mostrado na Figura 4. Já a segunda consiste em uma tela inicial de login, conforme mostra a Figura 5 que deve levar a uma tela de repositório de imagens armazenadas em banco de dados. Cada tarefa foi subdividida em quatro subtarefas, conforme descrito a seguir:

- ❏ Tarefa 1: Dado um modelo de cadastro de produtos de mercado, faça:
  - Alterar o tipo da variável setor de texto para *RadioButton*, contendo as opções "Açougue", "Hortifruti" e "Congelados".
  - Criar uma fila para ordenar os dados antes de inseri-los no banco de dados.
  - Criar uma tela que seja capaz de exibir os dados da tabela produto.
  - Recuperar os itens do banco de dados para mostrá-los na tela criada anteriormente.
- ❏ Tarefa 2: Dada uma tela de autenticação de usuário, faça:

- Limitar o usuário a formatar corretamente os dados de *login*, onde o usuário deve ser um CPF e a senha deve conter no mínimo 7 dígitos.
- Criar uma tela que seja capaz de exibir uma lista de imagens.
- Criar uma função que converta as imagens inseridas para um formato de entrada válido no banco de dados: para Python, usar as bibliotecas *OpenCV* e *pandas*; para Java, usar as bibliotecas *BufferedImage* e *SQLite*; e para Kotlin, usar as bibliotecas *bitmap* e *SQLite*.
- Criar um botão para excluir e outro para adicionar imagens ao repositório.

Após a definição das tarefas e a criação das implementações iniciais em cada uma das linguagens, foi realizada uma segunda etapa do estudo. Nesta fase, cada pesquisador ficou encarregado de resolver as tarefas originalmente criadas pelos outros colegas de equipe. Para isso, cada tarefa foi resolvida duas vezes: uma utilizando o Google como ferramenta de apoio para buscas e resolução de problemas, e outra utilizando o ChatGPT como assistente inteligente. Todas as sessões de resolução foram gravadas em vídeo para posterior análise.

A avaliação do desempenho das ferramentas foi adotada uma abordagem quantitativa baseada na análise detalhada das modificações realizadas nos códigos-fonte durante a execução de cada tarefa, semelhante ao referenciado na metodologia da tese (ROCHA, 2022). O objetivo foi mensurar o impacto e a complexidade das alterações realizadas pelos participantes ao utilizar cada uma das ferramentas de apoio (Google e ChatGPT) como suporte na resolução dos problemas propostos.

As atividades foram conduzidas com a versão 3.5 do ChatGPT e foram analisadas a inserção e alteração dos seguintes elementos de código: classes, métodos, variáveis, estruturas de repetição, estruturas de seleção, chamadas de métodos, bibliotecas e, no caso da linguagem Kotlin, arquivos gráficos XML. Também foi levado em consideração a quantidade de perguntas realizadas ao ChatGPT em comparação ao número de páginas acessadas no Google. A coleta e análise dos dados visam identificar a real eficácia do uso de assistentes baseados em inteligência artificial no contexto de manutenção de software, em comparação com os métodos tradicionais de pesquisa.

Também foram extraídos dados relevantes em relação ao tempo. Foram coletados o tempo total gasto para realizar cada tarefa, o tempo de manipulação do código no *Integrated Development Environment* (IDE) e o tempo de busca em cada uma das ferramentas, sendo que, para o ChatGPT, busca refere-se ao tempo de realização da pergunta e leitura da resposta fornecida e, para o Google, refere-se ao tempo de realização da consulta e à busca de informações nas páginas fornecidas. Por fim, decidiu-se extrair a relevância das soluções em três níveis distintos: relevante, parcialmente relevante e irrelevante. Uma solução para as tarefas do ChatGPT equivale a uma resposta fornecida pela IA, enquanto para o Google uma solução seria o mesmo que uma página navegada. E assim, temos

que as soluções relevantes são aquelas em que a dúvida é completamente solucionada. As parcialmente relevantes são as que não solucionam a dúvida mas ajudam parcialmente a entender a solução. Por fim, as irrelevantes são as páginas ou respostas que não indicam avanço algum à solução da dúvida.

3.2 Resultados

Nesta seção, serão apresentados os resultados obtidos a partir da utilização do ChatGPT e do Google nas tarefas descritas na Seção 3.1. A análise busca fornecer uma base empírica para as conclusões finais.

A seguir, são apresentadas as tabelas que mostram a quantidade de manipulação dos elementos técnicos de programação (como classes, métodos, variáveis etc.) nas resoluções das tarefas utilizando o ChatGPT e o Google para as três linguagens de programação consideradas neste estudo.

A Tabela 1 apresenta os resultados das tarefas realizadas em Java pelo Pesquisador 2. Observa-se que o número total de mudanças realizadas utilizando ChatGPT foi maior na Tarefa 1 (163 mudanças) em comparação ao Google (131 mudanças). Na Tarefa 2, o cenário se inverte, com o Google apresentando um número maior de mudanças (154) em relação ao ChatGPT (140). Aqui o ChatGPT apresenta maior número de chamadas de métodos alteradas, enquanto o Google se destaca em chamadas de métodos inseridas e o número de perguntas realizadas ao ChatGPT é menor do que o número de páginas acessadas no Google.

Tabela 1 – Tarefas em Java realizadas pelo Pesquisador 2 para análise quantitativa da complexidade da manutenção.

	Tarefa 1		**Tarefa 2**	
	ChatGPT	**Google**	**ChatGPT**	**Google**
Classes inseridas	2	2	3	3
Classes alteradas	3	1	1	2
Métodos inseridos	16	12	11	15
Métodos alterados	21	7	4	7
Variáveis inseridas	29	29	39	42
Variáveis alteradas	14	9	9	8
Estrutura de repetição inserida	2	2	0	0
Estrutura de repetição alterada	0	0	0	0
Estrutura de seleção inserida	3	3	6	7
Estrutura de seleção alterada	19	10	5	2
Chamada de métodos inserida	31	39	47	53
Chamada de métodos alterada	9	4	3	4
Bibliotecas adicionadas	14	13	12	11
Perguntas ao GPT / Páginas no Google	29	31	25	29
Mudanças totais	**163**	**131**	**140**	**154**

Na Tabela 2, os resultados do Pesquisador 3 em Java mostram uma distribuição mais

equilibrada entre ChatGPT e Google. Na Tarefa 1, o número total de mudanças é praticamente igual, levemente favorável ao ChatGPT (127 contra 130 do Google). Na Tarefa 2, o Google apresenta um número maior de mudanças (145 contra 133 de ChatGPT). O Google apresenta maior número de métodos inseridos e alterados na Tarefa 1, enquanto ChatGPT se destaca na Tarefa 2. O número de perguntas realizadas ao ChatGPT é menor do que o número de páginas acessadas no Google, especialmente na Tarefa 1.

Tabela 2 – Tarefas em Java realizadas pelo Pesquisador 3 para análise quantitativa da complexidade da manutenção.

	Tarefa 1		**Tarefa 2**	
	ChatGPT	**Google**	**ChatGPT**	**Google**
Classes inseridas	2	3	2	3
Classes alteradas	1	2	1	1
Métodos inseridos	13	17	9	16
Métodos alterados	9	6	14	7
Variáveis inseridas	28	30	37	39
Variáveis alteradas	8	3	11	15
Estrutura de repetição inserida	2	2	0	0
Estrutura de repetição alterada	2	0	0	0
Estrutura de seleção inserida	3	3	3	4
Estrutura de seleção alterada	11	6	5	8
Chamada de métodos inserida	36	39	41	37
Chamada de métodos alterada	2	8	1	4
Bibliotecas adicionadas	10	11	9	11
Perguntas ao GPT / Páginas no Google	8	26	21	34
Mudanças totais	**127**	**130**	**133**	**145**

Para Python, a Tabela 3 apresenta os resultados das tarefas realizadas pelo Pesquisador 1. O Google apresenta maior número total de mudanças em ambas as tarefas (138 e 179, respectivamente) em comparação ao ChatGPT (103 e 146). E, ao analisar cada elemento inserido individualmente, nota-se que o Google apresentou soluções mais complexas em todas as categorias. Quanto à quantidade de consultas, o número de perguntas realizadas ao ChatGPT é menor do que o número de páginas acessadas no Google.

A Tabela 4 apresenta os resultados das tarefas em Python realizadas pelo Pesquisador 2. O Google apresenta maior número total de mudanças em ambas as tarefas (160 e 192, respectivamente) em comparação ao ChatGPT (120 e 167). O Google apresenta maior número de métodos e variáveis inseridas em ambas as tarefas. Estruturas de seleção inseridas são mais frequentes com o Google. O Google apresenta maior número de chamadas de métodos inseridas e alteradas. O número de perguntas realizadas ao ChatGPT é menor do que o número de páginas acessadas no Google.

A Tabela 5 apresenta os resultados das tarefas realizadas pelo Pesquisador 1 na linguagem Kotlin. Observa-se que o Google apresenta maior número total de mudanças em ambas as tarefas (78 e 123, respectivamente) em comparação ao ChatGPT (69 e 113). O Google apresenta maior número de métodos e variáveis inseridas em ambas as tarefas. O

Tabela 3 – Tarefas em Python realizadas pelo Pesquisador 1 para análise quantitativa da complexidade da manutenção.

	Tarefa 1		**Tarefa 2**	
	ChatGPT	**Google**	**ChatGPT**	**Google**
Classes inseridas	0	0	0	0
Classes alteradas	1	1	1	1
Métodos inseridos	15	18	23	26
Métodos alterados	2	4	9	6
Variáveis inseridas	9	17	14	19
Variáveis alteradas	4	9	5	7
Estrutura de repetição inserida	1	3	0	0
Estrutura de repetição alterada	0	0	0	0
Estrutura de seleção inserida	2	3	9	12
Estrutura de seleção alterada	0	0	1	2
Chamada de métodos inserida	59	71	60	89
Chamada de métodos alterada	8	10	18	11
Bibliotecas adicionadas	2	2	6	6
Perguntas ao GPT / Páginas no Google	11	7	25	36
Mudanças totais	**103**	**138**	**146**	**179**

Tabela 4 – Tarefas em Python realizadas pelo Pesquisador 2 para análise quantitativa da complexidade da manutenção.

	Tarefa 1		**Tarefa 2**	
	ChatGPT	**Google**	**ChatGPT**	**Google**
Classes inseridas	0	2	0	0
Classes alteradas	1	3	1	1
Métodos inseridos	14	19	26	27
Métodos alterados	4	7	2	5
Variáveis inseridas	13	20	13	18
Variáveis alteradas	10	9	8	6
Estrutura de repetição inserida	1	0	0	0
Estrutura de repetição alterada	0	0	0	0
Estrutura de seleção inserida	1	3	13	18
Estrutura de seleção alterada	0	0	4	5
Chamada de métodos inserida	72	87	82	87
Chamada de métodos alterada	2	8	12	18
Bibliotecas adicionadas	2	2	6	7
Perguntas ao GPT / Páginas no Google	15	19	32	28
Mudanças totais	**120**	**160**	**167**	**192**

ChatGPT apresenta maior número de estruturas de repetição inseridas, enquanto o Google se destaca em estruturas de seleção. O número de perguntas realizadas ao ChatGPT é semelhante ao número de páginas acessadas no Google nesta amostra.

Por fim, na Tabela 6, os resultados do Pesquisador 3 em Kotlin mostram que o Google apresenta maior número total de mudanças em ambas as tarefas (133 e 153, respectivamente) em comparação ao ChatGPT (129 e 148). O Google apresenta maior número de métodos inseridos e alterados na Tarefa 1, enquanto o ChatGPT se destaca na Tarefa 2. O número de perguntas realizadas ao ChatGPT é maior na Tarefa 2, enquanto o Google

Tabela 5 – Tarefas em Kotlin realizadas pelo Pesquisador 1 para análise quantitativa da complexidade da manutenção.

	Tarefa 1		**Tarefa 2**	
	ChatGPT	**Google**	**ChatGPT**	**Google**
Classes inseridas	3	2	2	3
Classes alteradas	0	1	0	1
Métodos inseridos	4	2	10	13
Métodos alterados	0	1	4	6
Variáveis inseridas	11	15	23	20
Variáveis alteradas	10	9	13	11
Estrutura de repetição inserida	2	1	3	1
Estrutura de repetição alterada	0	0	0	0
Estrutura de seleção inserida	1	2	1	2
Estrutura de seleção alterada	0	0	0	0
Chamada de métodos inserida	29	35	42	51
Chamada de métodos alterada	2	4	8	9
Bibliotecas adicionadas	4	3	3	3
Arquivos gráficos XML adicionados	1	0	1	1
Arquivos gráficos XML alterados	2	3	3	2
Perguntas ao GPT / Páginas no Google	23	24	34	29
Mudanças totais	**69**	**78**	**113**	**123**

apresenta maior número de páginas acessadas na Tarefa 1.

Tabela 6 – Tarefas em Kotlin realizadas pelo Pesquisador 3 para análise quantitativa da complexidade da manutenção.

	Tarefa 1		**Tarefa 2**	
	ChatGPT	**Google**	**ChatGPT**	**Google**
Classes inseridas	1	3	2	3
Classes alteradas	1	1	2	1
Métodos inseridos	2	4	9	11
Métodos alterados	20	6	18	10
Variáveis inseridas	24	25	27	25
Variáveis alteradas	13	3	10	8
Estrutura de repetição inserida	2	0	4	2
Estrutura de repetição alterada	0	0	0	1
Estrutura de seleção inserida	3	2	2	2
Estrutura de seleção alterada	0	0	1	2
Chamada de métodos inserida	39	51	47	62
Chamada de métodos alterada	15	23	19	18
Bibliotecas adicionadas	5	3	3	3
Arquivos gráficos XML adicionados	1	4	1	1
Arquivos gráficos XML alterados	3	8	3	4
Perguntas ao GPT / Páginas no Google	44	22	30	53
Mudanças totais	**129**	**133**	**148**	**153**

Os resultados apresentados indicam que o Google frequentemente demonstra um maior número total de mudanças em comparação ao ChatGPT, especialmente em tarefas de maior complexidade. No entanto, para compreender os comportamentos gerais associados ao uso do Google e do ChatGPT, foram identificados padrões recorrentes em cada tarefa.

Esses padrões foram sintetizados na Tabela 10, que reúne as principais observações sobre as modificações realizadas, além de avaliar o desempenho das tarefas em termos de tempo de execução e relevância das soluções. Esses aspectos são detalhados nas Tabelas 7, 8 e 9. A relevância das buscas foi classificada em três níveis: relevante, parcialmente relevante e irrelevante. Adicionalmente, o tempo foi segmentado entre o uso da IDE e o tempo dedicado à busca por soluções, conforme descrito na seção 3.1.

Tabela 7 – Perguntas classificadas por relevância e análise do tempo gasto das interações dos Pesquisadores 2 e 3 com ChatGPT e Google durante a execução das tarefas em Java.

	Pesquisador 2				**Pesquisador 3**			
	Tarefa 1		**Tarefa 2**		**Tarefa 1**		**Tarefa 2**	
	ChatGPT	**Google**	**ChatGPT**	**Google**	**ChatGPT**	**Google**	**ChatGPT**	**Google**
Perguntas relevantes	17	8	15	7	4	7	10	8
Perguntas parcialmente relevantes	3	10	4	14	3	8	7	17
Perguntas irrelevantes	9	13	6	8	1	11	4	9
Tempo gasto no IDE	48m01s	56m12s	33m08s	46m31s	41m09s	41m34s	46m19s	51m25s
Tempo gasto no ChatGPT|Google	37m36s	1h34m49s	24m04s	58m57s	22m48s	1h14m37s	33m29s	1h04m46s
Diferença (IDE - ChatGPT|Google)	10m25s	-37m23s	9m04s	-12m26s	18m21s	-32m57s	12m50s	-13m21s
Tempo total da tarefa	**1h25m27s**	**2h31m01s**	**57m12s**	**1h45m28s**	**1h03m57s**	**1h56m11s**	**1h19m48s**	**1h56m11s**

Tabela 8 – Perguntas classificadas por relevância e análise do tempo gasto das interações dos Pesquisadores 1 e 3 com ChatGPT e Google durante a execução das tarefas em Kotlin.

	Pesquisador 1				**Pesquisador 3**			
	Tarefa 1		**Tarefa 2**		**Tarefa 1**		**Tarefa 2**	
	ChatGPT	**Google**	**ChatGPT**	**Google**	**ChatGPT**	**Google**	**ChatGPT**	**Google**
Perguntas relevantes	13	7	19	10	15	8	17	9
Perguntas parcialmente relevantes	6	5	9	5	11	3	4	24
Perguntas irrelevantes	4	12	6	14	18	11	9	20
Tempo gasto no IDE	27m42s	44m46s	48m57s	45m46s	1h22m22s	39m7s	1h34m16s	1h9m27s
Tempo gasto no ChatGPT|Google	16m11s	51m58s	36m37s	1h26m43s	23m18s	55m28s	40m58s	1h33m2s
Diferença (IDE - ChatGPT|Google)	11m31s	-7m32s	12m20s	-41m37s	59m04s	-16m21s	53m58s	-24m15s
Tempo total da tarefa	**43m53s**	**1h36m4s**	**1h25m34s**	**2h12m29s**	**1h45m40s**	**1h34m35s**	**2h15m14s**	**2h42m29s**

Tabela 9 – Perguntas classificadas por relevância e análise do tempo gasto das interações dos Pesquisadores 1 e 2 com ChatGPT e Google durante a execução das tarefas em Python.

	Pesquisador 1				**Pesquisador 2**			
	Tarefa 1		**Tarefa 2**		**Tarefa 1**		**Tarefa 2**	
	ChatGPT	**Google**	**ChatGPT**	**Google**	**ChatGPT**	**Google**	**ChatGPT**	**Google**
Perguntas relevantes	6	6	13	10	7	7	11	10
Perguntas parcialmente relevantes	3	4	4	9	4	2	14	10
Perguntas irrelevantes	2	7	8	17	4	10	7	8
Tempo gasto no IDE	30m03s	24m04s	24m36s	47m10s	37m13s	47m43s	42m11s	50m56s
Tempo gasto no ChatGPT|Google	17m24s	38m28s	39m07s	53m02s	14m01s	28m17s	25m18s	51m09s
Diferença (IDE - ChatGPT|Google)	12m39s	-14m24s	-14m31s	-5m52s	23m12s	19m26s	16m57s	-13s
Tempo total da tarefa	**47m27s**	**1h2m32s**	**1h3m43s**	**1h40m12s**	**51m14s**	**1h16m**	**1h7m23s**	**1h42m05s**

De forma geral, as Tabelas 7, 8 e 9 revelam que o tempo total das tarefas realizadas com o Google tende a ser maior em comparação ao ChatGPT, devido ao maior tempo gasto na busca por soluções, além de apresentar um número maior de páginas acessadas em comparação com o número de perguntas realizadas ao ChatGPT, o que pode indicar uma abordagem mais exploratória. Por outro lado, o ChatGPT apresenta maior eficiência

em termos de tempo, com menor duração total das tarefas e maior proporção de perguntas classificadas como relevantes. Essas diferenças refletem características distintas das ferramentas: enquanto o Google oferece uma ampla gama de informações, o ChatGPT tende a fornecer respostas mais diretas e específicas, reduzindo o tempo necessário para encontrar soluções. Com isso, foi possível extrair os achados da Tabela 10.

Tabela 10 – Tabela de Achados Gerais da pesquisa

Achado	Título	Descrição
1	Tempo de IDE em tarefas do ChatGPT em razão do tempo total da tarefa.	Ao usar o ChatGPT, a porcentagem do tempo gasto na IDE em relação ao tempo total de resolução do problema é, em média, de 61,87%.
2	Tempo de acesso a páginas em tarefas do Google em razão do tempo total da tarefa.	Ao usar o Google, a porcentagem do tempo gasto acessando páginas e buscando soluções é, em média, de 53,51% em relação ao tempo total da tarefa.
3	Diferença de páginas acessadas no Google e perguntas realizadas no ChatGPT.	Ao comparar a diferença entre a quantidade de páginas acessadas no Google e perguntas realizadas no ChatGPT, o Google apresentou aproximadamente 12,8% mais consultas por tarefa que o ChatGPT.
4	Complexidade baseada em quantidade de elementos inseridos, alterados e removidos.	A complexidade do código produzido com o auxílio do Google é maior, considerando a quantidade de elementos inseridos, alterados e removidos no código final. As soluções fornecidas pelo Google foram 1,44% mais complexas do que as soluções geradas pelo ChatGPT.
5	Quantidade de alterações realizadas em cada tarefa.	Analisando as variáveis de elementos alterados, o ChatGPT tende a apresentar valores maiores. As soluções do ChatGPT possuem 2,6% mais alterações. do que as soluções fornecidas com o auxílio do Google.
6	Diferença do tempo total gasto em tarefas.	O tempo total gasto em tarefas com o Google é aproximadamente 35 minutos e 44 segundos maior do que o tempo total gasto em tarefas com o ChatGPT.
7	Diferença na quantidade de soluções relevantes.	Por tarefa, o ChatGPT apresentou 37,9% mais soluções relevantes do que as apresentadas pelo Google.
8	Diferença na quantidade de soluções parcialmente relevantes.	Em relação a soluções parcialmente relevantes, temos que o Google apresenta 31,7% soluções parcialmente relevantes a mais do que o ChatGPT, em média.
9	Diferença na quantidade de soluções irrelevantes.	As páginas do Google apresentaram 46,3% soluções irrelevantes a mais do que o ChatGPT.

O Achado 1 indica que uma parte significativa do tempo total de resolução de problemas com o ChatGPT é dedicada ao desenvolvimento e implementação de código diretamente na IDE, correspondendo a uma média de 61,87% do tempo total. Isso sugere que o ChatGPT fornece respostas e soluções que exigem menos tempo de pesquisa e mais tempo de aplicação prática, permitindo que os desenvolvedores permaneçam focados na codificação e ajustem o código de forma mais direta. A eficiência na busca de soluções relevantes oferecidas pelo ChatGPT pode ser um fator que contribui para essa maior dedicação ao tempo de trabalho prático na IDE, refletindo uma abordagem mais integrada ao fluxo de desenvolvimento.

Ainda na Tabela 10, o Achado 2 revela que 53,51% do tempo total de resolução de problemas, utilizando o Google, é dedicado à busca por informações e soluções *online*. Isso sugere que uma parte significativa do processo de resolução de problemas envolve a navegação por diversas fontes, como artigos, fóruns e tutoriais, para encontrar respostas ou exemplos relevantes. Essa alta porcentagem de tempo gasto na pesquisa indica que o uso do Google pode exigir mais esforço e tempo para filtrar informações úteis e aplicáveis, resultando em uma menor proporção de tempo dedicado à implementação direta de soluções na IDE. Assim, o processo de busca com o Google pode ser mais disperso, demandando mais tempo de exploração e comparação de diversas fontes para chegar a uma solução satisfatória.

O Achado 3 mostra que os usuários realizam 12,8% mais consultas no Google do que no ChatGPT para completar uma tarefa. Esse resultado indica que, ao usar o Google, os usuários tendem a explorar mais páginas para encontrar informações relevantes, o que pode estar relacionado à necessidade de buscar, comparar e validar diversas fontes para chegar a uma solução confiável. Por outro lado, o menor número de perguntas feitas ao ChatGPT sugere que o modelo é capaz de fornecer respostas mais diretas e abrangentes, reduzindo a necessidade de múltiplas consultas. Isso evidencia que o ChatGPT pode oferecer um caminho mais eficiente na obtenção de informações, concentrando o processo de resolução de problemas em um único ponto de consulta.

O Achado 4 revela que o código gerado com o auxílio do Google tende a ser mais complexo. As soluções obtidas por meio do Google contêm 1,44% elementos a mais em comparação com aquelas geradas com o ChatGPT. Isso sugere que o uso do Google pode levar a soluções com tamanho de código levemente maior, possivelmente devido à necessidade de adaptar múltiplas referências ou integrar diferentes pedaços de informação para formar uma solução completa. Por outro lado, o ChatGPT parece oferecer soluções mais diretas e simplificadas, com menos alterações e elementos adicionais, o que pode resultar em um código mais limpo e fácil de manter. Esse contraste indica que o ChatGPT tende a fornecer respostas mais focadas e otimizadas a longo prazo, enquanto o Google pode proporcionar uma abordagem mais fragmentada e laboriosa na construção de soluções.

O Achado 5 indica que as soluções geradas pelo ChatGPT tendem a exigir um maior número de elementos alterados em comparação com aquelas obtidas pelo Google. Ao observar as linhas de classes, métodos, variáveis, estruturas de repetição e seleção e chamadas de métodos alteradas, separadamente em relação às inseridas, notou-se que as soluções do ChatGPT possuem 2,6% alterações a mais do que as soluções baseadas no Google. Isso sugere que, embora o ChatGPT possa fornecer respostas diretas e concisas, elas podem necessitar de ajustes mais frequentes para se adaptar aos requisitos específicos do problema. Esse resultado pode refletir a natureza interativa do ChatGPT, onde os usuários ajustam e refinam as respostas fornecidas para atender melhor às suas necessidades. Em contraste, as soluções obtidas por meio do Google, por serem geralmente baseadas

em exemplos já aplicados e discutidos em diversos contextos, podem necessitar de menos ajustes diretos, embora ainda possam ser mais complexas em termos de quantidade total de elementos.

O Achado 6 revela que, em média, o uso do Google para resolver problemas consome cerca de 35 minutos e 44 segundos a mais do que o uso do ChatGPT, o que representa 30,9% mais tempo. Isso indica que, ao utilizar o Google, os usuários podem gastar um tempo considerável navegando por múltiplas páginas, verificando a relevância das informações e integrando dados de diferentes fontes. Por outro lado, o ChatGPT parece oferecer um processo mais eficiente, com respostas mais diretas e focadas, o que reduz o tempo necessário para chegar a uma solução. Essa diferença significativa no tempo de resolução sugere que o ChatGPT pode ser uma ferramenta mais ágil e prática para desenvolvedores que buscam resolver problemas de forma rápida e eficiente, sem precisar explorar extensivamente várias fontes de informação.

O Achado 7 mostra que o ChatGPT oferece 37,9% soluções relevantes a mais por tarefa em comparação com o Google. Isso indica que o ChatGPT tem uma maior capacidade de fornecer respostas úteis e diretamente aplicáveis às necessidades dos usuários, o que pode reduzir o tempo gasto em pesquisa e aumentar a eficiência na resolução de problemas. Em contraste, o Google pode apresentar uma variedade maior de informações, mas nem todas são imediatamente relevantes ou aplicáveis, exigindo que os usuários filtrem e validem as respostas manualmente. Esse resultado reforça a vantagem do ChatGPT em oferecer soluções de alta qualidade de forma mais concentrada, simplificando o processo de busca e aplicação de conhecimento.

O Achado 8 indica que, em relação a soluções parcialmente relevantes, o Google apresenta 31,7% soluções parcialmente relevantes a mais do que o ChatGPT. Essa diferença sugere que, quando se trata de respostas que são apenas parcialmente úteis, o Google tende a oferecer um número ligeiramente maior de soluções que podem necessitar de ajustes ou de uma combinação com outras fontes para se tornarem completamente aplicáveis, reforçando a necessidade de uma combinação de soluções distintas ao usar o Google.

O Achado 9 revela que as páginas acessadas pelo Google apresentam 46,3% soluções irrelevantes a mais do que as respostas fornecidas pelo ChatGPT. Isso sugere que o uso do Google para resolver problemas pode resultar em um volume maior de informações que não são úteis ou aplicáveis, obrigando os usuários a gastar mais tempo filtrando conteúdos desnecessários. Em contraste, o ChatGPT tende a fornecer respostas mais focadas e pertinentes, com uma menor incidência de informações irrelevantes. Esse resultado destaca uma vantagem significativa do ChatGPT na capacidade de oferecer soluções mais qualificadas e direcionadas, o que pode aumentar a eficiência do processo de busca e a produtividade dos usuários ao evitar distrações com conteúdos fora do contexto ou sem utilidade prática.

A Tabela 10 sintetiza os principais achados relacionados ao desempenho do ChatGPT

e do Google na resolução de tarefas, destacando diferenças significativas em termos de tempo, relevância das soluções e complexidade das respostas. De forma geral, os resultados indicam que o ChatGPT apresenta maior eficiência temporal, com menor tempo médio de execução das tarefas e maior proporção de soluções relevantes, enquanto o Google, embora mais demorado, oferece uma abordagem mais exploratória, com maior número de consultas parcialmente relevantes e maior complexidade nas soluções geradas.

Capítulo 4

Conclusão

O presente estudo teve como objetivo principal avaliar a efetividade do modelo de linguagem generativa ChatGPT 3.5 em comparação com o motor de busca Google no suporte a desenvolvedores durante a execução de tarefas de manutenção de *software*. A pesquisa foi conduzida por meio de um estudo envolvendo três pesquisadores, que realizaram tarefas funcionalmente equivalentes nas linguagens Java, Kotlin e Python, utilizando ambas as ferramentas como suporte.

Os resultados demonstraram que o ChatGPT apresentou maior capacidade de fornecer respostas contextualmente adequadas e diretamente aplicáveis, o que se refletiu em menor tempo médio de busca e menor número de consultas necessárias para alcançar soluções satisfatórias. Embora, em certos casos, tenha demandado mais modificações em trechos de código preexistente, as soluções geradas apresentaram menor complexidade estrutural, com um número reduzido de elementos inseridos, sugerindo maior objetividade e precisão.

Adicionalmente, os dados empíricos indicaram que o uso do ChatGPT favoreceu a reutilização de código, otimizando o tempo de implementação e promovendo maior concentração dos participantes no ambiente de desenvolvimento IDE. Em contraste, o Google, embora amplamente informativo, exigiu maior esforço de triagem por parte dos participantes, resultando em maior tempo de navegação, o que impactou negativamente a eficiência do processo. Os achados sugerem que o ChatGPT configura-se como uma ferramenta eficaz no apoio à manutenção de sistemas, especialmente em atividades que exigem compreensão e modificação de código existente.

4.1 Principais Contribuições

Entre as principais contribuições deste trabalho, destacam-se a identificação de padrões comportamentais distintos na interação com ferramentas baseadas em inteligência artificial e motores de busca tradicionais, bem como a apresentação de evidências quantitativas sobre a complexidade e o tempo envolvidos na execução das tarefas.

A pesquisa contribui para o avanço da Engenharia de *Software* ao oferecer uma análise comparativa entre duas das ferramentas mais utilizadas por desenvolvedores na atualidade. Além da avaliação detalhada das modificações realizadas no código, o estudo fornece subsídios relevantes sobre a eficiência das ferramentas, o comportamento dos usuários e a aplicabilidade de modelos de linguagem natural em cenários práticos de manutenção de *software*.

4.2 Trabalhos Futuros

Como desdobramento deste estudo, sugere-se a ampliação do número de participantes e de tarefas analisadas, bem como a inclusão de novas ferramentas baseadas em IA, como o *GitHub Copilot* ou o *Claude*. Outra possibilidade de trabalho futuro consiste na avaliação qualitativa da experiência dos desenvolvedores com as ferramentas, investigando aspectos subjetivos como satisfação, confiança e percepção de utilidade. Ademais, investigações mais profundas sobre os tipos de erros cometidos por modelos generativos e estratégias de mitigação também podem enriquecer o campo.

Referências

ANAGNOSTOPOULOS, C.-N. Chatgpt impacts in programming education: A recent literature overview that debates chatgpt responses. **University of Aegean, Cultural Technology and Communication Dpt, Intelligent Systems lab, Lesvos, Mytilene 81100, Greece**, 2024. Citado na página 10.

DANTAS, C.; ROCHA, A.; MAIA, M. Assessing the readability of chatgpt code snippet recommendations: A comparative study. In: **Proceedings of the XXXVII Brazilian Symposium on Software Engineering**. New York, NY, USA: Association for Computing Machinery, 2023. (SBES '23), p. 283–292. ISBN 9798400707872. Disponível em: <https://doi.org/10.1145/3613372.3613413>. Citado na página 22.

EBERT, C.; LOURIDAS, P. Generative ai for software practitioners. **IEEE Software**, v. 40, n. 4, p. 30–38, July 2023. ISSN 1937-4194. Citado na página 22.

GOMES, A.; HENRIQUES, J.; MENDES, A. A. J. A. Uma proposta para ajudar alunos com dificuldades na aprendizagem inicial de programação de computadores. **Educação, Formação e Tecnologias**, scielo, v. 01, p. 93 – 103, 05 2008. ISSN 1646-933x. Disponível em: <http://educa.fcc.org.br/scielo.php?script=sci_arttext&pid=S1646-933x2008000100009&nrm=iso>. Citado na página 11.

HAROON, S.; C.A., H.; A.S., J. Generative pre-trained transformer (gpt) based model with relative attention for de novo drug design. **Computational Biology and Chemistry**, v. 106, p. 107911, 2023. ISSN 1476-9271. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1476927123001020>. Citado na página 19.

HELM, J. et al. Machine learning and artificial intelligence: Definitions, applications, and future directions. **Current Reviews in Musculoskeletal Medicine**, v. 13, p. 69–76, 2020. Disponível em: <https://doi.org/10.1007/s12178-020-09600-8>. Citado na página 18.

JIN, K. et al. Can chatgpt support developers? an empirical evaluation of large language models for code generation. In: **Proceedings of the 21st International Conference on Mining Software Repositories**. New York, NY, USA: Association for Computing Machinery, 2024. (MSR '24), p. 167–171. ISBN 9798400705878. Disponível em: <https://doi.org/10.1145/3643991.3645074>. Citado na página 24.

KRUEGER, C. W. Software reuse. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 24, n. 2, p. 131–183, jun 1992. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/130844.130856>. Citado na página 15.

LIDDY, E. D. Natural language processing. In: **Encyclopedia of Library and Information Science**. 2nd. ed. New York, NY: Marcel Dekker, Inc., 2001. Citado na página 17.

LIENTZ, B. P.; SWANSON, E. B.; TOMPKINS, G. E. Characteristics of application software maintenance. **Communications of the ACM**, ACM, v. 21, n. 6, p. 466–471, June 1978. Citado na página 15.

MICROSOFT. **Microsoft Teams Up with OpenAI to Exclusively License GPT-3 Language Model**. 2020. <https://blogs.microsoft.com/blog/2020/09/22/microsoft-teams-up-with-openai-to-exclusively-license-gpt-3-language-model/>. Accessed: February 14, 2024. Citado na página 10.

MIN, E. et al. Transformer for graphs: An overview from architecture perspective. **arXiv preprint arXiv:2202.08455**, 2022. Citado na página 19.

MORATIS, K. et al. Write me this code: An analysis of chatgpt quality for producing source code. In: . New York, NY, USA: Association for Computing Machinery, 2024. (MSR '24), p. 147–151. ISBN 9798400705878. Disponível em: <https://doi.org/10.1145/3643991.3645070>. Citado na página 24.

NUNES, H. et al. Evaluating the effectiveness of llms in fixing maintainability issues in real-world projects. **Proceedings of SANER 2025**, 2025. Disponível em: <https://arxiv.org/abs/2502.02368>. Citado na página 23.

RAHMANIAR, W. Chatgpt for software development: Opportunities and challenges. Institute of Electrical and Electronics Engineers (IEEE), ago. 2023. Disponível em: <http://dx.doi.org/10.36227/techrxiv.23993583.v1>. Citado na página 21.

RAJBHOJ, A. et al. Accelerating software development using generative ai: Chatgpt case study. In: **Proceedings of the 17th Innovations in Software Engineering Conference (ISEC 2024)**. ACM, 2024. p. 5:1–5:11. Disponível em: <https://dl.acm.org/doi/abs/10.1145/3641399.3641403>. Citado na página 23.

ROCHA, A. M. **Mineração de soluções relevantes para tarefas de programação a partir de resultados de mecanismos de busca**. Tese (Doutorado) — Universidade Federal de Uberlândia, Uberlândia, Brasil, 2022. Disponível em: <https://doi.org/10.14393/ufu.te.2022.565>. Citado 2 vezes nas páginas 22 e 28.

ROCHA, A. M.; MAIA, M. A. Mining relevant solutions for programming tasks from search engine results. **IET Software**, v. 17, n. 4, p. 455–471, 2023. Disponível em: <https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/sfw2.12127>. Citado 2 vezes nas páginas 11 e 22.

RUSSO, D. Navigating the complexity of generative ai adoption in software engineering. **ACM SIGSOFT Software Engineering Notes**, Association for Computing Machinery, New York, NY, USA, v. 47, n. 5, p. 135–140, set. 2023. ISSN 1049-331X. Disponível em: <https://doi.org/10.1145/3652154>. Citado na página 23.

SEYMOUR, T.; FRANTSVOG, D.; KUMAR, S. History of search engines. **International Journal of Management Information Systems (IJMIS)**, v. 15, n. 4, p. 47–58, 2011. Disponível em: <https://doi.org/10.19030/ijmis.v15i4.5799>. Citado na página 15.

SHANAHAN, M. Talking about large language models. **Communications of the ACM**, v. 67, n. 2, p. 68–79, 2024. Citado na página 18.

SHUVO, U. A. et al. Assessing chatgpt's code generation capabilities with short vs long context programming problems. In: **Proceedings of the 11th International Conference on Networking, Systems, and Security**. New York, NY, USA: Association for Computing Machinery, 2025. (NSysS '24), p. 32–40. ISBN 9798400711589. Disponível em: <https://doi.org/10.1145/3704522.3704535>. Citado na página 25.

SOUZA, S. C. B. d.; ANQUETIL, N.; OLIVEIRA, K. M. d. Which documentation for software maintenance? **Journal of the Brazilian Computer Society**, Sociedade Brasileira de Computação, v. 12, n. 3, p. 31–44, Dec 2006. ISSN 0104-6500. Disponível em: <https://doi.org/10.1007/BF03194494>. Citado na página 14.

TOPAL, M. O.; BAS, A.; HEERDEN, I. van. **Exploring Transformers in Natural Language Generation: GPT, BERT, and XLNet**. 2021. Citado na página 21.

VASWANI, A. et al. **Attention Is All You Need**. 2023. Disponível em: <https://arxiv.org/abs/1706.03762>. Citado 3 vezes nas páginas 6, 19 e 20.

WILLS, R. S. Google's pagerank: The math behind the search engine. **North Carolina State University**, May 2006. Disponível em: <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=33566b740d3cd0c0dde57e13b5da148bef37376f>. Citado 3 vezes nas páginas 6, 16 e 17.

XIAO, T. et al. Devgpt: Studying developer-chatgpt conversations. In: **2024 IEEE/ACM 21st International Conference on Mining Software Repositories (MSR)**. [S.l.: s.n.], 2024. p. 227–230. Citado na página 24.

XU, R.; FENG, Y.; CHEN, H. **ChatGPT vs. Google: A Comparative Study of Search Performance and User Experience**. 2023. Disponível em: <https://arxiv.org/abs/2307.01135>. Citado na página 24.

ZHANG, M.; LI, J. A commentary of gpt-3 in mit technology review 2021. **Fundamental Research**, KeAi Publishing, 2021. Disponível em: <http://www.keaipublishing.com/en/journals/fundamental-research/>. Citado na página 21.