UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Lorena Carla Jordão Braga Vilaça

Uso de LLM para análise de vieses de gênero em ferramentas de controle de versão

Recife

2025

Lorena Carla Jordão Braga Vilaça

Uso de LLM para análise de vieses de gênero em ferramentas de controle de versão

Trabalho apresentado ao programa de Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Área de Concentração: Engenharia de Software

Orientador: Kiev Santos de Gama

Coorientadora: Gabriela Cunha Sampaio

Recife

2025

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Vilaça, Lorena Carla Jordão Braga.
Uso de LLM para análise de vieses de gênero em ferramentas de controle de versão / Lorena Carla Jordão Braga Vilaça. - Recife, 2025.
58 p. : il., tab.

Orientador(a): Kiev Santos da Gama
Cooorientador(a): Gabriela Cunha Sampaio
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Informática, Ciências da Computação - Bacharelado, 2025.
Inclui referências, apêndices.

1. Llama3. 2. Análise de tom. 3. LLMs. 4. Viés de gênero. 5. Open-source. I. Gama, Kiev Santos da. (Orientação). II. Sampaio, Gabriela Cunha. (Coorientação). IV. Título.

000 CDD (22.ed.)

LORENA CARLA JORDÃO BRAGA VILAÇA

USO DE LLM PARA ANÁLISE DE VIESES DE GÊNERO EM FERRAMENTAS DE CONTROLE DE VERSÃO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de bacharel em Ciência da Computação.

Aprovado em: 14/08/2025

BANCA EXAMINADORA

_____

Prof. Dr. Kiev Santos de Gama (Orientador)

Universidade Federal de Pernambuco

_____

Profa. Dra. Paola Rodrigues de Godoy Accioly (Examinador Interno)

Universidade Federal de Pernambuco

_____

Dra. Gabriela Cunha Sampaio (Examinador Externo)

Meta

AGRADECIMENTOS

É fundamental reconhecer as pessoas que contribuíram para a realização deste projeto e para a minha trajetória acadêmica e profissional. Sem o apoio delas, este trabalho não teria sido concluído.

Agradeço ao meu pai, Júnior Vilaça, por sempre fazer tudo ao seu alcance para me proporcionar uma boa educação, desde pequena. À minha mãe, Cristina Vilaça, agradeço não só pelo esforço incansável em me apoiar na trajetória acadêmica, mas também por ser um verdadeiro modelo de mulher e determinação.

Às minhas irmãs, minhas melhores amigas e minhas maiores inspirações. Lívia Vilaça, que foi a razão da minha escolha pela Ciência da Computação, meu principal suporte nos momentos difíceis da vida e meu exemplo de coragem e força em um ambiente de trabalho muitas vezes difícil. Luiza Vilaça, meu exemplo constante de dedicação e esforço, que sempre encontrava tempo para me incentivar com tarefinhas de casa depois da escola. É por causa de vocês que me tornei a mulher que sou hoje, e por isso serei eternamente grata.

Aos meus sobrinhos, Letícia Rocha, Rafael Rocha, Alice Rocha e Mateus Bezerra, que foram fontes de leveza, carinho, distração e apoio nos momentos mais difíceis deste processo. Vocês mantêm viva a minha criança interior.

À Beatrix Lee, minha namorada, pelas noites em claro me dando suporte e incentivo, por ser quase uma terceira orientadora, e por representar um eterno carinho e conforto para mim nos momentos difíceis do projeto.

Agradeço aos meus amigos de graduação, em especial a Elaine Cruz, Guilherme Macêdo e Thaís Couto, que tornaram esses anos de graduação mais especiais. Sou muito grata pelos momentos e alegrias compartilhadas ao longo dessa jornada.

Ao meu amigo Lu Barbosa, pelo apoio, incentivo, revisões e dicas durante todo o processo. Este trabalho não seria o mesmo sem o seu suporte.

À Apple Developer Academy, por me mostrar que a computação vai muito além de programar, que cada linha de código tem impacto na vida de alguém. Obrigada por me apresentar pessoas que levarei para a vida toda.

Por fim, agradeço aos meus orientadores, Gabriela Sampaio e Kiev Gama, pela paciência e orientação ao longo de todo o processo, por me apontarem caminhos possíveis diante das dificuldades do projeto e pelo constante apoio.

RESUMO

O estudo tem como objetivo avaliar o uso de modelo de aprendizagem de grande escala (Large Language Model — LLM) para identificar e analisar potenciais vieses de gênero no ambiente de desenvolvimento open-source, com ênfase nas diferenças linguísticas observadas em pull requests do GitHub. A metodologia envolve a coleta e o pré-processamento de dados por meio da API do GitHub, a inferência de gênero a partir do nome próprio dos usuários, a classificação das mensagens utilizando o modelo Llama 3 e, por fim, a avaliação dos resultados. A pesquisa contribui para a compreensão de padrões linguísticos associados ao gênero em comunidades open-source, fornecendo material para criação de ambientes mais inclusivos na engenharia de software.

Palavras-chaves: Llama3; Análise de tom; LLMs; Viés de gênero; Open-source.

ABSTRACT

The study aims to evaluate the use of a large-scale learning model (Large Language Model — LLM) to identify and analyze potential gender biases in the open-source development environment, with an emphasis on linguistic differences observed in GitHub pull requests. The methodology involves collecting and preprocessing data through the GitHub API, inferring gender from users' first names, classifying messages using the Llama 3 model, and finally evaluating the results. The research contributes to the understanding of gender-associated linguistic patterns in open-source communities, providing material for the creation of more inclusive environments in software engineering.

Keywords: Llama3; Tone Analysis; LLMs; Gender bias; Open-source.

LISTA DE FIGURAS

LISTA DE CÓDIGOS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O desenvolvimento de software open-source é um ecossistema colaborativo que, idealmente, deveria avaliar as contribuições com base exclusivamente em critérios técnicos. No entanto, pesquisas recentes (WANG; REDMILES, 2019; CAVALCANTE, 2020; MAY; WACHS; HANNAK, 2019; ZOLDUOARRATI; LICORISH, 2021) apontam para a presença de *vieses de gênero* [1] nesse ambiente, revelando que, na prática, outros fatores podem influenciar a aceitação das contribuições. Esses estudos mostram que desenvolvedoras mulheres enfrentam desafios específicos, como taxas de aceitação de *pull requests* significativamente mais baixas em comparação aos homens, mesmo quando a qualidade técnica é equivalente.

Plataformas como o GitHub (2025), muito utilizadas para controle de versionamento e colaboração em projetos de software, oferecem um grande conjunto de dados para investigar possíveis causas desse fenômeno. O GitHub hospeda projetos de diferentes tamanhos e níveis de popularidade, desde bibliotecas mantidas por grandes empresas até repositórios criados por desenvolvedores independentes. Além disso, disponibiliza uma interface específica para *pull requests*, que permite que qualquer pessoa proponha modificações em um projeto, possibilitando tanto a análise das contribuições técnicas quanto das interações interpessoais envolvidas no processo.

Há trabalhos anteriores que investigaram padrões de comunicação no contexto da engenharia de software, como os de Ahmed e Srivastava (2017) e Zolduoarrati e Licorish (2021), que analisaram interações no Stack Overflow. Esses estudos indicam que homens e mulheres tendem a adotar estilos de comunicação distintos, mesmo em ambientes técnicos. Os resultados mostraram que desenvolvedores do gênero masculino tendem a adotar uma linguagem mais competitiva, autoritária, autocentrada e orientada a conquistas, enquanto desenvolvedoras do gênero feminino demonstram uma linguagem mais coletiva, com maior ênfase em aspectos

[1] O viés de gênero é a produção e reprodução de estereótipos, que reforçam assimetrias sociais entre gêneros. Como descreve Pereira et al. (2024) ao analisar resultados de um modelo gerativo de imagens "investigando os resultados do modelo gerativo de imagens Dall-E mini[7], mostram a presença de estereotipia que expande e intensifica os desbalanços já presentes na sociedade." O autor exemplifica alguns desses vieses ao dizer "em relação a gênero, o trabalho mostra: a) uma tendência geral a gerar resultados tidos como masculinos; b) a propensão a resultados homogeneamente masculinos ou femininos - isto é, uma tendência a gerar imagens somente tidas como masculinas ou femininas, quando a distribuição da ocupação pelos órgãos institucionais estadunidenses de estatísticas laborais preveem resultados mais equilibrados -; e c) a reprodução de estereótipos. Como exemplos deste, podemos citar tanto a reafirmação de profissões majoritariamente ocupadas por mulheres (secretária/o, cabeleireira/o, maquiador/a, recepcionista, nutricionista) e homens (piloto, construtor/a, minerador/a, eletricista, encanador/a), mas também a inversão tanto de posições majoritariamente masculinas, representadas como majoritariamente femininas (vendedor/a, apresentador/a e locutor/a de notícias, cantor/a) e vice e versa (garçom/nete, padeira/o, contador/a".

sociais, positivos e colaborativos, contribuindo para um ambiente mais inclusivo e construtivo.

Diferentemente do Stack Overflow (2025), que se concentra em perguntas e respostas, o ambiente colaborativo do GitHub permite correlacionar diretamente as interações entre desenvolvedores com contribuições concretas, como os *pull requests*. Essa relação entre comunicação e aceitação de contribuições pode fornecer indícios de vieses de gênero implícitos, e ainda há uma lacuna na literatura quanto à análise dessas diferenças em ambientes de colaboração por *pull requests*.

Esse cenário dificulta a pluralidade e limita a participação do gênero feminino em ambientes de desenvolvimento open-source. Portanto, é fundamental investigar métodos capazes de identificar e quantificar esses possíveis vieses de gênero de forma mais precisa, explorando novas abordagens que permitam compreender melhor como eles se manifestam nas interações entre desenvolvedores, para que o direcionamento dos esforços de combate a esse problema seja aplicados de forma otimizada.

Dessa forma, o presente trabalho busca investigar a eficácia do uso de Large Language Models (LLMs) para identificação de vieses de gênero, por meio de análise de interações textuais em *pull requests*, na plataforma GitHub. Para isso, foram coletados dados de 237.040 *pull requests* utilizando a API oficial do GitHub, totalizando 646.452 mensagens entre comentários e revisões. A inferência de gênero dos usuários foi realizada por meio do modelo LLaMA 3, disponibilizado pela plataforma Ollama, que classificou os nomes próprios dos usuários, enquanto a classificação do tom das mensagens foi conduzida utilizando o modelo meta-llama/Meta-Llama-3-8B-Instruct, implementado via Hugging Face (Hugging Face, 2025a).

Os resultados obtidos indicam uma predominância significativa de usuários classificados no gênero masculino, refletindo a sub-representação feminina na plataforma. A análise do tom das mensagens revelou diferenças sutis na forma de comunicação entre gêneros, entretanto, a confiabilidade da classificação automatizada se mostrou baixa, o que impediu conclusões definitivas sobre vieses na comunicação.

A pesquisa está organizada da seguinte forma: na Seção 2 são abordados os conceitos básicos necessários para o desenvolvimento da pesquisa, incluindo uma visão geral sobre LLMs, seus modelos de análise de tom de texto, o LLaMA 3, bem como conceitos relacionados ao GitHub, *pull requests* e o viés de gênero no ambiente open-source. Na Seção 3, é descrita a metodologia utilizada, contemplando as etapas de coleta e pré-processamento dos dados, o processo de inferência de gênero e a classificação de tom. A Seção 4 apresenta os resultados obtidos e suas respectivas análises. Por fim, a Seção 5 traz a conclusão do trabalho, seguida

das considerações finais, limitações encontradas e sugestões para trabalhos futuros.

Para fins deste estudo, adotamos uma perspectiva de gênero baseada na diferença entre "masculino" e "feminino". Reconhecemos que o gênero constitui um espectro socialmente construído e não se limita ao modelo binário. Como apontado por Piscitelli (2009, p. 5), "sexo está vinculado à biologia (hormônios, genes, sistema nervoso e morfologia) e gênero tem relação com a cultura (psicologia, sociologia, incluindo aqui todo aprendizado vivido desde o nascimento)", o que demonstra que as noções e definições de gênero vão além da vinculação ao sexo genital. No entanto, a adoção de uma abordagem binária neste estudo se justifica pela ausência de informações explícitas sobre identidade de gênero na plataforma analisada. Assim, optamos por uma análise quantitativa entre "masculino" e "feminino" como um primeiro passo frente à falta de dados, com o objetivo de construir conhecimento sobre possíveis desigualdades no contexto do desenvolvimento de software open-source.

1.1 OBJETIVOS

Esta pesquisa tem como objetivo utilizar o modelo de linguagem LLama3 para identificar possíveis vieses de gênero em ambientes de desenvolvimento open-source no GitHub, a partir da categorização de mensagens em categorias pré-selecionadas. O foco principal é a análise de características linguísticas presentes em comentários e revisões de *pull requests*, comparando contribuições realizadas por desenvolvedores dos gêneros masculino e feminino.

2 CONCEITOS BÁSICOS

2.1 LARGE LANGUAGE MODELS

LLMs são modelos treinados em uma grande quantidade de parâmetros e dados textuais, permitindo que desempenhem tarefas variadas como análise semântica, geração de conteúdos e tradução de textos. Devido ao grande volume de dados em que foram treinados, os LLMs são capazes de aprender padrões linguísticos complexos e gerar respostas semelhantes às humanas, permitindo aplicações diversas como análise de sentimentos, monitoramento de mídias sociais e interpretação de interações escritas (BOMMASANI et al., 2021). Isso representa um grande avanço no campo do processamento de linguagem natural (PLN), área que foca seus estudos na capacitação dos computadores a entender, interpretar e gerar linguagem humana, tanto escrita quanto falada.

Como consequência dessa praticidade de resposta, os LLMs se tornaram famosos em contextos de análise social, pois permitem a interpretação automatizada de interações escritas em mídias sociais. Essa tecnologia já é utilizada por empresas para monitorar opiniões de consumidores, detectar padrões de comportamento e avaliar sentimentos em escala.

2.1.1 Modelos de análise de tom de texto

A análise de tom é uma tarefa de PLN que busca identificar a intenção ou emoção expressa em um texto. Tradicionalmente, esse tipo de análise era feita com modelos supervisionados treinados com os rótulos: positivo, negativo ou neutro. No entanto, com o avanço da tecnologia e dos modelos de aprendizagem de máquina, vem se tornando possível realizar análises de tom mais robustas. Já existem estudos que mostram a eficácia de utilizar modelos de linguagem de ponta para detectar automaticamente comentários em tom de sugestão em fóruns de cursos online (SÁNCHEZ et al., 2024).

Ao permitir a customização por meio de prompts (instruções criadas em linguagem natural), os LLMs podem ampliar as possibilidades da análise de tom para além de classificações rígidas. Ao invés de classificar apenas nos três rótulos citados anteriormente, seria possível escolher categorias diversas, que, reforçadas por frases de exemplo, permiriam ao modelo adaptar sua interpretação ao contexto específico da tarefa. Essa flexibilidade torna os LLMs úteis em ambientes onde os comentários podem assumir funções variadas e não se encaixam facilmente

em rótulos genéricos de sentimento.

2.1.2 LLaMA 3

O modelo utilizado neste trabalho é o LLaMA 3 (Large Language Model Meta AI), um LLM desenvolvido pela Meta AI. O LLaMA 3 apresenta uma evolução em relação às versões anteriores, com melhorias em desempenho, capacidade de generalização e compreensão contextual.

A versão utilizada localmente foi integrada por meio da plataforma Ollama (2025), que permite a execução local de LLMs. O LLaMA 3 apresenta desempenho competitivo de compreensão textual e geração de linguagem, além de permitir adaptação a tarefas específicas por meio de *fine-tuning* ou *prompt engineering*. Como modelo de código aberto, ele constitui uma alternativa viável aos modelos fechados amplamente utilizados no mercado, como o GPT-4o, sendo avaliado positivamente em benchmarks como MMLU e ARC (META, 2024).

2.2 GITHUB

Segundo documentação oficial, o GitHub é uma plataforma baseada em nuvem em que é possível armazenar, compartilhar e trabalhar com outras pessoas para escrever códigos (GITHUB, 2025). Essa plataforma se tornou uma das principais ferramentas utilizadas diariamente por desenvolvedores(as) em todo o mundo para trabalhar em projetos de software de forma colaborativa.

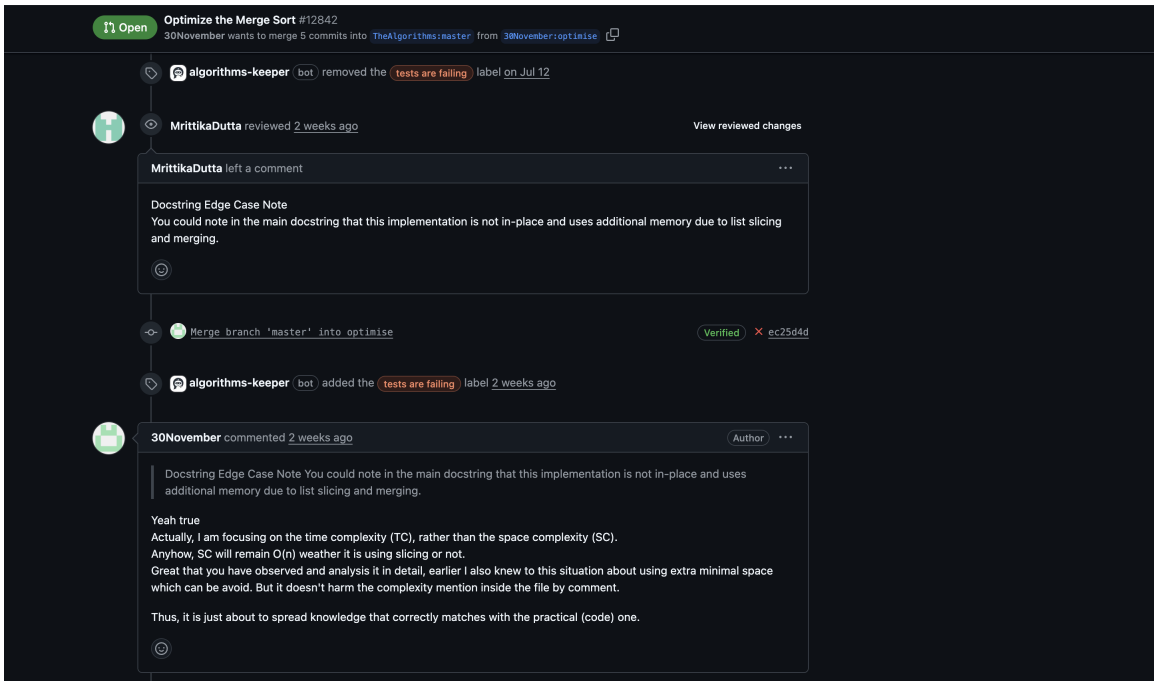
Além de armazenar repositórios de código, é possível controlar permissões, rastrear novos problemas através de issues e documentar. No contexto do open-source, o GitHub oferece uma possibilidade muito rica de obter a contribuição de pessoas diversas através de *pull requests*.

2.2.1 Pull request

O *Pull Request* (PR) é uma das funcionalidades principais no fluxo de trabalho do GitHub. Ele permite que colaboradores proponham alterações em um projeto de forma organizada, submetendo as modificações feitas em seu *branch* [2] para revisão e, eventualmente, integradas ao código principal do repositório.

[2] Tradução livre para o português da palavra branch é "ramo"

Figura 1 – Seção de discussão da interface de um PR no Github



Fonte: Pull request realizado no projeto TheAlgorithms/Python no Github (2025)

Ao criar um PR, o autor descreve as mudanças propostas, justificando suas razões e fornecendo contexto para os demais contribuidores daquele repositório. Cada PR cria um espaço de interação, onde outros usuários podem revisar o código, sugerir melhorias, aprovar ou solicitar ajustes antes da integração final. Esse processo favorece a qualidade do código, sendo adotado em projetos de engenharia de software, principalmente os de código aberto.

De acordo com H.urna (2020), o ciclo de vida de um pull request segue as seguintes etapas:

1. Criação do *Branch* e Desenvolvimento: Um desenvolvedor cria um *branch* separado para trabalhar em uma nova funcionalidade ou corrigir um erro. As alterações são feitas localmente e depois enviadas para o repositório remoto.

2. Abertura do *Pull Request*: Ao concluir o desenvolvimento, o desenvolvedor abre um PR, comparando o *branch* de origem com o *branch* principal (geralmente o main ou master). Nesse momento, é incluída uma informação de descrição sobre as mudanças realizadas.

3. Revisão de Código: Os revisores analisam o PR. Eles podem deixar comentários técnicos ou recomendações de padrões de estilo daquele repositório, levantar questionamentos, sugerir modificações ou aprovar as mudanças. Esse processo pode envolver múltiplas iterações entre autor e revisores.

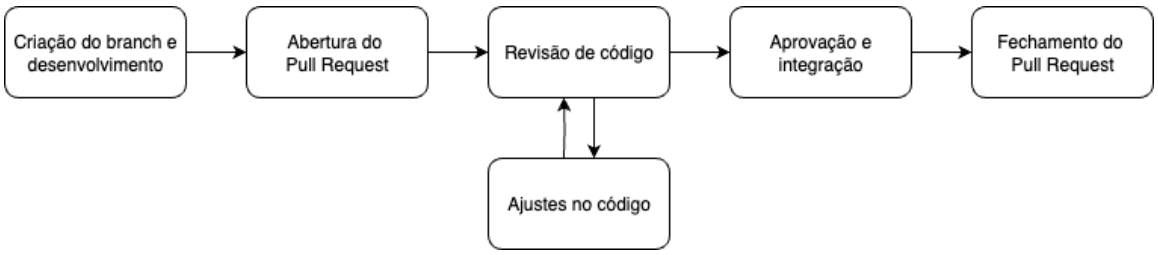
4. Ajustes: O autor do PR pode realizar novos *commits* no mesmo *branch* para alterar o código baseado nas sugestões recebidas. O GitHub atualiza automaticamente o PR com as alterações mais recentes.

5. Aprovação e Integração: Uma vez que os revisores aprovam o PR, ele pode ser aceito no branch principal do projeto.

6. Fechamento: O PR é encerrado. Em alguns casos, ele pode ser fechado sem ser aceito, caso as alterações não sejam consideradas adequadas ou já tenham sido incorporadas de outra forma.

Esse fluxo de trabalho promove a qualidade técnica e abre um amplo espaço para estudo dos aspectos sociais no ambiente do desenvolvimento open-source do GitHub.

Figura 2 – Diagrama do ciclo de vida de um Pull Request



Fonte: Imagem elaborada pela autora (2025)

2.3 VIÉS DE GÊNERO NO OPEN-SOURCE

A participação de mulheres no desenvolvimento de software open-source tem sido limitada em comparação à dos homens. Como descrevem Vasilescu et al. (2015, p. 9): "Atualmente, entretanto, mulheres programadoras estão em minoria no Open Source Software e outros times técnicos."

Além disso, Terrell et al. (2017) demonstraram que as contribuições feitas por mulheres possuem, em média, maior taxa de aceitação que as de homens, mas apenas quando o gênero da autora não é identificável. Quando o perfil do contribuinte se torna público como sendo de uma mulher, a taxa de aceitação diminui, sugerindo a presença de viés de gênero no processo de revisão de código. Esses resultados indicam que, embora a qualidade técnica das contribuições femininas seja equiparável ou superior, a percepção de demarcadores de gênero pode afetar negativamente sua aceitação e visibilidade.

Outro ponto apontado na literatura são os padrões e tendências identificados no tom da linguagem dos gêneros. No estudo de Newman et al. (2008), observou-se que as mulheres eram mais propensas a usar frases verbais de incerteza, como "eu suponho"e "eu imagino", para reduzir a força de suas declarações. Já Zolduoarrati e Licorish (2021) revelou que as desenvolvedoras femininas expressavam elogios e gratidão com mais frequência do que os desenvolvedores masculinos no Stack Overflow, e que esses exibiram uma atitude mais individualista, usando a palavra "eu"com mais frequência do que as mulheres.

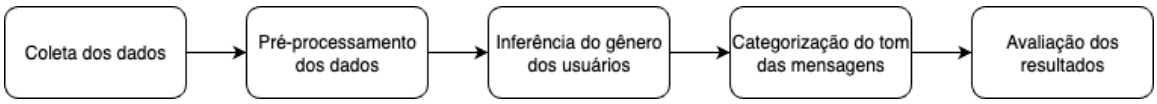
Além da taxa de aceitação de PRs, a literatura observa outros indicadores que evidenciam desigualdade de gênero. Por exemplo, em projetos open-source, a participação de mulheres em revisões de código é menor em comparação com os homens, frequentemente relacionado ao baixo recebimento de convites para revisão destinados a pessoas do gênero feminino. Em 11 dos 14 casos estudados por Sultana, Turzo e Bosu (2022) as mulheres têm uma participação significativamente menor em revisão de código que os outros homens com quem trabalham.

Apesar de haver avanços no debate sobre diversidade em tecnologia, há uma lacuna na análise das interações interpessoais que ocorrem dentro dos PRs, como comentários e revisões. Poucos estudos exploram, por exemplo, se o tom dos comentários varia de acordo com o gênero da pessoa que submeteu o PR ou da pessoa que revisa. Diante desse cenário, o trabalho busca investigar se é possível observar o viés de gênero na comunicação textual dos PRs, por meio da análise de comentários e revisões presentes no GitHub.

3 METODOLOGIA

Esta pesquisa foi realizada em cinco etapas: coleta de *pull requests* do Github, pré-processamento do conjunto de dados, inferência de gênero baseada no nome do usuário, categorização do tom das mensagens e avaliação dos resultados obtidos.

Figura 3 – Diagrama de fluxo da pesquisa



Fonte: Imagem elaborada pela autora (2025)

3.1 COLETA E PRÉ PROCESSAMENTO DOS DADOS

Os dados utilizados nesta pesquisa foram obtidos por meio da API do GitHub, com foco em repositórios de código aberto. Foram selecionados 10 repositórios com maior número de estrelas, métrica que representa os favoritos da plataforma, para cada uma das 10 linguagens de programação mais populares de janeiro de 2024, segundo o ranking publicado por O'Grady (2024). As linguagens escolhidas foram: JavaScript, Python, Java, PHP, C#, TypeScript, CSS, C++, Ruby e C. A lista completa dos repositórios selecionados consta em Apêndice A

Foram coletados os *pull requests* desses repositórios referentes aos últimos 5 anos ou, quando o projeto foi criado mais recentemente, desde sua origem. As informações extraídas incluíram: nome do repositório, ID do PR, título, status de *merge* [3], estado atual do PR, além de listas com comentários e revisões associadas.

Dentro dessas listas, foram registrados o nome e o username do autor do comentário ou revisão, o conteúdo textual da mensagem e a data de envio. A coleta foi realizada inteiramente por meio da API pública do GitHub, utilizando *endpoints* para busca de usuários, repositórios mais populares por linguagem e recuperação de comentários e revisões de cada PR.

Já durante o processo de coleta foi aplicado um filtro para remover interações realizadas por *bots*, comuns em tarefas automatizadas como testes ou atualização de dependências. Para isso, foram descartados os dados de usuários cujo campo tipo (`type`) era `"Bot"` ou cujo username terminava com `"bot"`. A listagem B apresenta a versão resumida do script

[3] O status de merge indica se uma alteração realizada em um PR foi integrada ou não ao projeto

utilizado para a coleta dos pull requests. Foram implementadas funções auxiliares para lidar com paginação da API, autenticação, exclusão de bots, obtenção de comentários e revisões, e salvamento dos resultados em arquivos JSON e CSV.

O conjunto de dados inicial reuniu 237.040 PRs, dos quais 166.843 (70,39%) estavam com status *merged*. Os demais estavam distribuídos entre 9.031 PRs abertos (3,81%) e 61.166 PRs fechados sem *merge*. No total, foram extraídos 373.301 comentários e 511.238 revisões.

A próxima etapa foi uma verificação para remover possíveis repetições de repositórios, uma vez que um projeto pode estar associado a mais de uma linguagem. Os 10 repositórios coletados nas linguagens C++, C e C# acabaram sendo os mesmos, por isso, dos 30 que foram coletados, foi mantida apenas uma versão de cada.

Em seguida, foram mantidos apenas os dados de PRs em que o campo de nome do usuário estava preenchido. Durante a coleta, identificou-se que 34,31% dos comentários (128.088 registros) e 21.52% das revisões (109.999 registros) não possuíam o campo de nome (`name`) preenchido. Esse campo, embora opcional na criação de um perfil no GitHub, é essencial para esta pesquisa por permitir a inferência de gênero a partir do nome.

Como o foco do estudo é a análise das interações interpessoais entre gêneros, o conjunto final reúne apenas os 646.452 comentários e revisões extraídas desses PRs. Um resumo do conjunto pode ser visto na Tabela 1.

Tabela 1 – Totais de comentários e revisões com nome preenchido

Categoria	**Quantidade**
Comentários com nome	245.213
Revisões com nome	401.239
Total geral	**646.452**

3.2 INFERÊNCIA DE GÊNERO

Após a coleta e o processamento dos dados, foi necessário inferir o gênero de cada usuário. O GitHub não fornece um campo específico para autoidentificação de gênero, por isso a forma escolhida para essa classificação foi a inferência a partir do nome próprio. Essa abordagem segue o método utilizado por Vasilescu, Capiluppi e Serebrenik (2013), que propuseram a inferência de gênero com base no nome e, quando disponível, no país de origem do usuário.

> O nome de uma pessoa costuma indicar seu gênero. Por exemplo, John é um nome masculino comum em inglês, enquanto Claire é um nome feminino

> comum em inglês. Combinado com informações de localização, é possível fazer inferências ainda mais precisas sobre o gênero de uma pessoa com base em seu nome. (VASILESCU; CAPILUPPI; SEREBRENIK, 2013)

A ferramenta inicialmente escolhida para essa tarefa foi a API Genderize.io (2025a), especializada na inferência de gênero com base no primeiro nome. Essa API faz suas previsões baseadas na proporção entre pessoas do gênero masculino e do feminino que possuem aquele nome, a partir de um banco de dados de mais de um bilhão de pessoas Genderize.io (2025b). Contudo, o limite diário de 100 requisições gratuitas mostrou-se insuficiente para a demanda, sendo necessário buscar uma solução alternativa.

A alternativa adotada foi o uso de LLM para realizar a inferência. O modelo foi instruído por meio de um *prompt* a considerar aspectos como origem cultural e linguística, terminações típicas por gênero, frequência histórica de uso e variações regionais. A classificação retornava três categorias: masculino, feminino ou desconhecido. Esta última foi destinada a nomes amplamente utilizados por ambos os gêneros, como "Taylor". A execução foi feita localmente com o modelo LLaMA3, por meio da plataforma Ollama.

Antes da inferência, os nomes passaram por um pré-processamento. Foram removidos espaços excedentes, caracteres não alfabéticos e os nomes foram convertidos para letras minúsculas, o que evitou que nomes repetidos fossem classificados mais de uma vez. Também foi aplicada uma filtragem para verificar a validade dos nomes, excluindo aqueles com menos de três caracteres e removendo nomes compostos apenas por dígitos, utilizando a função `isdigit()` da linguagem Python.

Para manipulação eficiente dos dados, todos os nomes foram extraídos da base e organizados em um dicionário, associando cada nome ao gênero atribuído. Esse procedimento permitiu reaproveitar as classificações em diferentes etapas da análise, evitando chamadas redundantes ao modelo.

Os hiperparâmetros utilizados para a execução do modelo foram `temperature=0,1` e `top_p=0,9`. A temperatura baixa (0,1) torna o modelo menos aleatório e mais determinístico, isso ajuda a reduzir ruído e erros inesperados, já que a tarefa exige classificação e não geração livre de texto. Já o parâmetro `top_p=0,9`, que fala sobre *nucleus sampling* (amostragem do núcleo), permite que o modelo selecione tokens dentro dos 90% de probabilidade acumulada, garantindo um equilíbrio entre controle e flexibilidade. Dessa forma, o modelo evita se prender ao token mais provável, mas também limita escolhas improváveis que poderiam comprometer a classificação. A listagem C apresenta a versão resumida do script utilizado

para a classificação de gênero dos nomes. No código completo, foram adicionados mecanismos de tolerância a falhas, tratamento de exceções e logging.

Código Fonte 1 – Prompt utilizado para a classificação de gênero a partir de nomes próprios.

```
You are a name classification expert. Based on your knowledge of names in global
    cultures and languages, classify the likely gender associated with the
    following name.

        If the name is commonly associated with males, return "male".
        If it is commonly associated with females, return "female".
        If the name is ambiguous, not a real name, or has no typical gender,
            return "unknown".

        Respond ONLY in valid JSON like this:
        {{"gender": "male"}}

        Examples:
        - "Carlos" -> {{"gender": "male"}}
        - "Fatima" -> {{"gender": "female"}}
        - "Taylor" -> {{"gender": "unknown"}}
        - "aabb" -> {{"gender": "unknown"}}

        Classify the name: {name}
```

Como resultado, foi gerado um arquivo no formato JSON contendo objetos com os atributos *"name"* e *"gender"*, como representado na Figura 4.

Figura 4 – Dicionário de nomes

Fonte: Arquivo gerado pela autora (2025)

Para avaliar a acurácia, 10 nomes foram selecionados aleatoriamente e classificados com a ferramenta Genderize. Em seguida, os resultados obtidos foram comparados aos do modelo, como mostra a Tabela 2. Observa-se que, em todos os casos, o modelo apresentou a mesma classificação do Genderize.

Tabela 2 – Comparação da classificação manual e do Llama3

Nome	Resultado Genderize	Probabilidade Genderize	Resultado Llama3
geetoormvn	unknown	–	unknown
stephan	male	100%	male
aviral	male	100%	male
raju	male	97%	male
mannat	female	92%	female
ravin	male	88%	male
evert	male	99%	male
haydn	male	97%	male
cezar	male	99%	male
elizabeth	female	100%	female

Fonte: Elaborada pela autora (2025)

A etapa final da inferência consistiu em cruzar o dicionário de nomes e gêneros com o conjunto de dados que continha as mensagens. Dessa forma, obteve-se uma base consolidada, contendo as informações essenciais para a análise proposta pela pesquisa, como será demonstrado mais afundo na seção 4.1.

3.3 CLASSIFICAÇÃO DE TOM

O processo de classificação das mensagens foi realizado por meio de uma série de iterações e testes, com o objetivo de aprimorar a qualidade dos resultados obtidos.

A etapa inicial de classificação consistiu na escolha das categorias a serem utilizadas. As categorias foram definidas tendo como base o trabalho de Zolduoarrati e Licorish (2021, p. 8), que realizou uma análise de viés de gênero no Stack Overflow, empregando tanto um método baseado em script quanto um método de codificação manual. Como essa análise foi feita em outro ambiente, foi necessária uma adaptação para adequar as categorias ao contexto do GitHub. As principais alterações ocorreram nas descrições de cada categoria, além da unificação de algumas, como as categorias *"type 1 question"* e *"type 2 question"* que se tornaram apenas *"Question"*. Além disso, visando observar se padrões identificados nos estudos de mencionados

em 2.3 estavam presentes, foram adicionadas as categorias de "incerteza" e "agradecimento". A lista com as categorias utilizadas na primeira iteração pode ser observada na Tabela 3.

Tabela 3 – Categorias de mensagens com suas descrições e exemplos

Nº	Categoria	Descrição	Exemplo
1	Question	Requests information due to a knowledge gap	"Why was this line removed?"
2	Answer	Provides a direct response to a question or problem	"This function is used to sanitize input from the form."
3	Information Exchange	Shares relevant info from other parts of the project or team	"This bug was already fixed in PR #421."
4	Discussion	Expresses thoughts or opinions about the code or design	"This design seems more flexible in the long term."
5	Approval	Indicates explicit approval of the code or pull request	"LGTM"
6	Reflection	Self-assessment or retrospective insight	"In hindsight, the way we handled error logging could be improved."
7	Scaffolding/Suggestion	Advice or encouragement, often collaborative and constructive	"Maybe we should split this into smaller components."
8	Instruction/Command	Directive or request for change, but stated in a collaborative or polite tone	"Please add tests before merging."
9	Bossy/Authoritarian	Directive expressed in a demanding or forceful tone	"Fix this now."
10	Gratitude	Thanks or praise	"Thanks for fixing this!"
11	Uncertainty	Suggestion or comment with hesitant or uncertain tone	"I'm not sure, but you could try this way."
12	Off-task	Irrelevant to the work item	"Going on vacation tomorrow."
13	Apology	Expressing regret	"Sorry, I pushed the wrong commit."
14	Not coded	Communications that cannot be assigned to any other category	"Lorem ipsum dolor sit amet..."

Devido à alta demanda computacional do modelo utilizado, o processamento foi realizado em um *cluster* disponibilizado pelo Centro de Informática da Universidade Federal de Pernambuco. O Cluster Apuana é uma infraestrutura de alto desempenho baseada em GPU (Graphics

Processing Unit), desenvolvida para suportar pesquisas na área de Inteligência Artificial, especialmente *Deep Learning*. Atualmente, o *cluster* conta com 10 nodes de processamento, totalizando 11 GPUs RTX3090 e 5 GPUs A100, além de 5 TB de memória RAM, 464 núcleos e 928 *threads* de processamento (CIN, s.d).

Para a instalação do modelo na infraestrutura do *cluster*, foi utilizada a plataforma Hugging Face (2025a), que é uma ferramenta online voltada ao desenvolvimento e compartilhamento de recursos para inteligência artificial, com foco em aprendizado de máquina e processamento de linguagem natural. O modelo selecionado foi o Meta-Llama-3-8B-Instruct do conjunto meta-llama, pré-treinado em mais de 15 trilhões de tokens públicos e ajustado para tarefas de conversação (Hugging Face, 2025b).

Com as categorias definidas e o modelo instalado e configurado, foi realizada a primeira tentativa de classificação na base de dados completa. A utilização da técnica de prompt de poucos disparos (*few-shot prompting*) permite o aprendizado no contexto, onde fornecemos um exemplos de classificação no prompt para direcionar o modelo para um melhor desempenho. A técnica pode ser observada no prompt listado abaixo:

Código Fonte 2 – Prompt utilizado para a classificação de tom das mensagens.

```
You are an expert in analyzing GitHub repository communications. Your task is to classify the following message into exactly one of the 14 categories below.

Categories with Examples:
1. Question: Requests information due to a knowledge gap
Example: "Why was this line removed?"

2. Answer: Provides a direct response to a question or problem
Example: "This function is used to sanitize input from the form."

3. Information Exchange: Shares relevant info from other parts of the project or team
Example: "This bug was already fixed in PR #421."

4. Discussion: Expresses thoughts or opinions about the code or design
Example: "This design seems more flexible in the long term."

5. Approval: Indicates explicit approval of the code or pull request
Example: "LGTM" or "Approved."

6. Reflection: Self-assessment or retrospective insight
Example: "In hindsight, the way we handled error logging could be improved."

```

```
    7. Scaffolding / Suggestion: Advice or encouragement, often collaborative and
        constructive
23  Example: "Maybe we should split this into smaller components."

25  8. Instruction / Command: Directive or request for change, but stated in a
        collaborative or polite tone
    Example: "Please add tests before merging."
27
    9. Bossy / Authoritarian: Directive expressed in a demanding or forceful tone
29  Example: "Fix this now."

31  10. Gratitude: Thanks or praise
    Example: "Thanks for fixing this!"
33
    11. Uncertainty: Suggestion or comment with hesitant or uncertain tone
35  Example: "I'm not sure, but you could try this way."

37  12. Off-task: Irrelevant to the work item
    Example: "Going on vacation tomorrow."
39
    13. Apology: Expressing regret
41  Example: "Sorry, I pushed the wrong commit."

43  14. Not coded: Communications that cannot be assigned to any other category
    Example: "Lorem ipsum dolor sit amet..."
45
    Message to classify: "{}"
47
    Instructions:
49      - Read the message carefully
        - Compare it with the examples provided
51      - Consider the intent and tone
        - Choose only ONE category that best fits
53      - Respond with ONLY the category number (1-14)

55  Classification:
```

Após a primeira classificação, observou-se que havia uma concentração em algumas categorias, em especial na categoria de sugestão. Esse comportamento foi identificado mesmo em mensagens que não possuíam conteúdo claro e intenção definida e deveriam ser classificadas como *"Not Coded"*, como mostrado na Tabela 5:

O erro encontrado mostrava que a configuração inicial do prompt e a definição das categorias não estavam bem ajustadas ao contexto do conjunto de mensagens analisado. Por isso, foi feita uma nova etapa exploratória, usando um conjunto menor de dados (10 mensagens)

Tabela 5 – Exemplo de mensagens categorizadas erradas

Mensagem	**Categoria**
English teacher	7 - Suggestion
* Duplicate of #2550* Missing explanation	7 - Suggestion
Pinging @elastic/es-docs (Team:Docs)	7 - Suggestion

para testes. Nessa fase, foram avaliadas diferentes variações de categorias, incluindo a retirada de algumas que não eram tão relevantes para os objetivos e a reformulação das descrições para evitar ambiguidades. Também foram testadas mudanças na estrutura e no formato do prompt, como a adição ou remoção de exemplos.

Com base nos resultados, foi definida uma nova lista de categorias (ver Apêndice I) e adotada a estratégia de *zero-shot prompting*, em que o modelo não recebe exemplos de classificação no próprio prompt. As categorias foram ajustadas a fim de reduzir redundâncias, sendo removida a categoria *discussion*, devido à sua grande semelhança com *suggestion*, e a categoria *off-task*, por transmitir a mesma ideia de *not coded*. Essa nova configuração foi aplicada a todo o conjunto de dados.

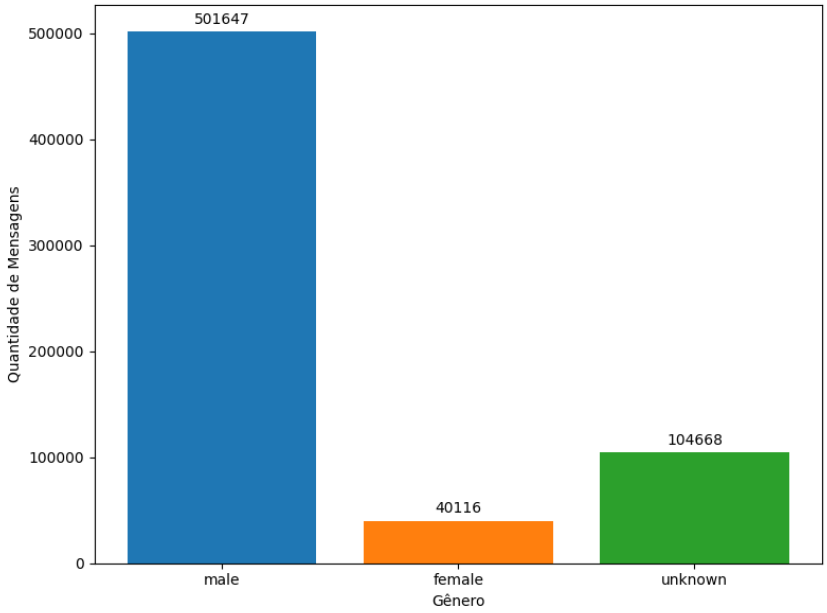
Por fim, para verificar a acurácia dessa segunda classificação, foi feita uma validação manual. Para isso, um conjunto aleatório de 200 mensagens, 100 do gênero feminino e 100 do masculino, foi classificado manualmente e comparado com as categorias atribuídas pelo modelo, seguindo a mesma metodologia usada na validação da inferência de gênero. Como forma de evitar viés, no momento dessa classificação as mensagens foram isoladas do gênero e lidas individualmente.

4 RESULTADOS E ANÁLISE

4.1 INFERÊNCIA DE GÊNERO DOS USUÁRIOS

A classificação de gênero realizada identificou que aproximadamente 77,61% dos usuários foram classificados no gênero masculino, 6,21% no feminino e 16,19% permaneceram como gênero desconhecido devido à ambiguidade ou ausência de dados suficientes. Ao todo, foram extraídos 14.227 nomes únicos para a construção do dicionário de classificação. A Figura 5 mostra em números absolutos como essa divisão se apresenta.

Figura 5 – Distribuição do conjunto de dados por gênero



Fonte: Gráfico elaborado pela autora (2025)

A validação comparando 10 nomes selecionados aleatoriamente com a API Genderize mostrou uma alta correção entre os métodos, reforçando a confiabilidade do uso do LLM para esta tarefa.

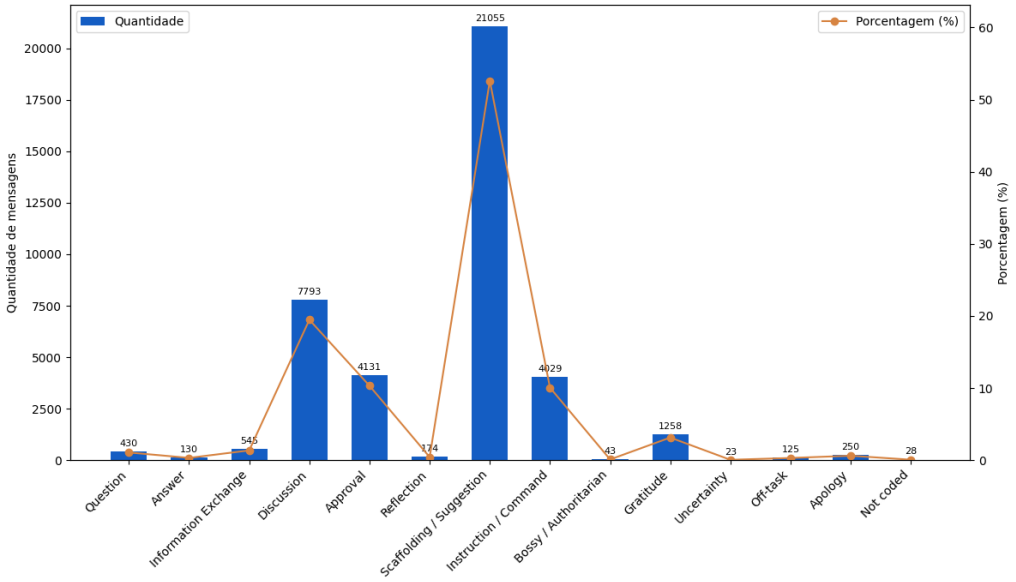
É possível perceber uma grande disparidade no volume da classificação, com predominância de usuários do gênero masculino em relação aos femininos, o que confirma o que já foi mostrado em outros estudos sobre a sub-representação feminina em ambientes de desenvolvimento de software (QIU et al., 2023). Nesse estudo, dos 1.823.414 usuários filtrados que contribuíram para projetos de código aberto, 911.990 (50,02%) foram identificados como homens e apenas 54.859 (3,01%) como mulheres.

4.2 CLASSIFICAÇÃO DO TOM DAS MENSAGENS

A primeira classificação do tom das mensagens resultou numa classificação enviesada para a categoria de sugestão. Em ambos os gêneros essa categoria concentrou mais de 50% das mensagens classificadas, com valores absolutos muito superiores às demais classes.

Observa-se que, na primeira classificação de tom, a categoria de sugestão concentrou mais da metade das mensagens em ambos os gêneros. Esse comportamento sugere um viés do modelo para interpretar mensagens como sugestões ou orientações, possivelmente devido ao ambiente de colaboração do Github. Como visto em 3.3, após análise exploratória dessa classificação, foram observados vários erros dentro dessa categoria, o que motivou etapas de iteração para refinamento na classificação.

Figura 6 – Primeira classificação de tom - Gênero feminino

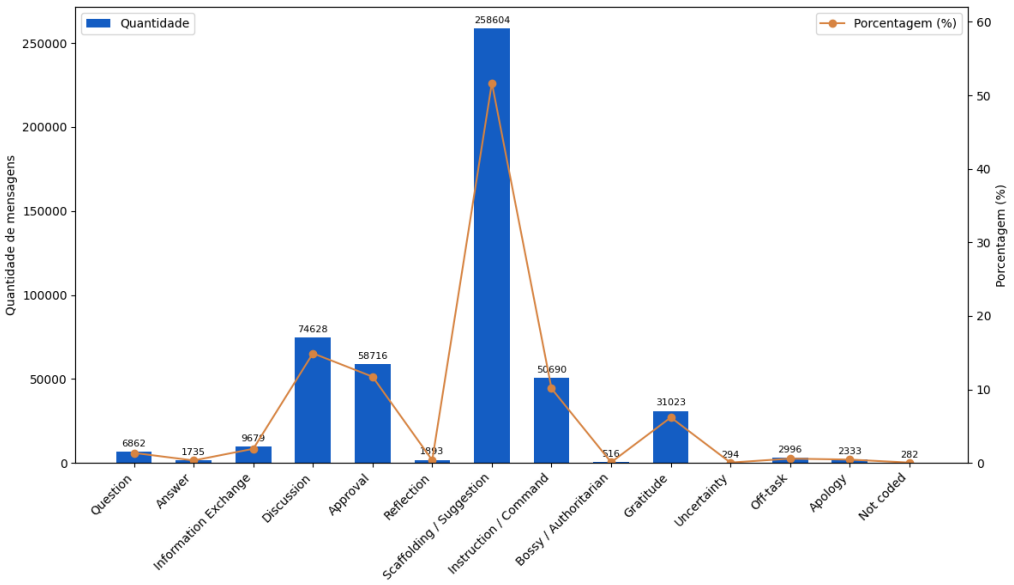


Fonte: Gráfico elaborado pela autora (2025)

A transição do modelo de *few-shot prompting* para *zero-shot prompting*, com ajuste das categorias, contribuiu para a redução da classificação excessiva de mensagens como "Sugestão"e melhorou a precisão geral da categorização, como evidenciado nas Figura 8 e Figura 9

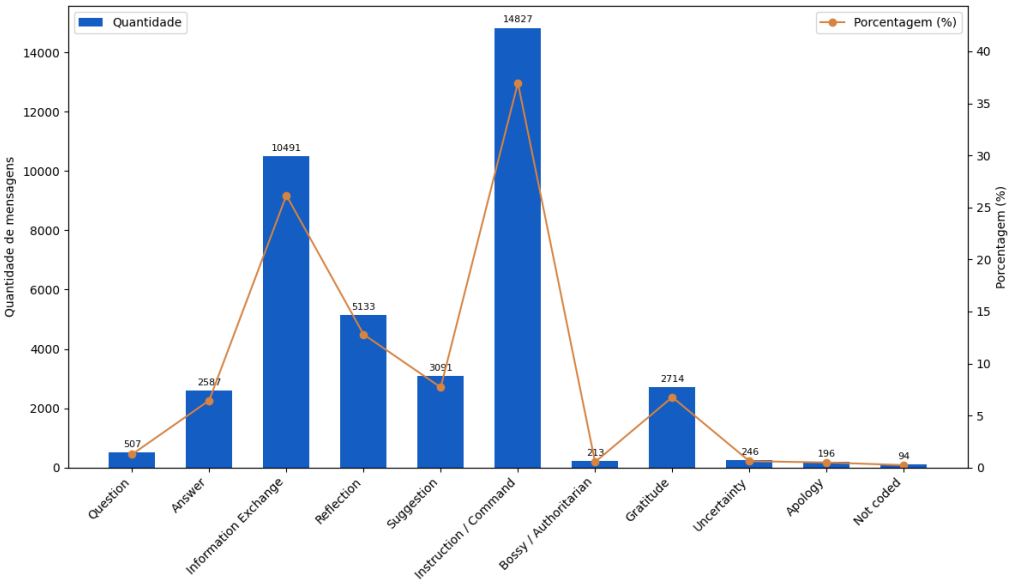
Na segunda classificação, observou-se uma grande redistribuição das categorias, com a de instrução se tornando a mais frequente para ambos os gêneros, representando aproximadamente 40% das mensagens analisadas. As categorias de troca de informação, reflexão e agradecimento surgem em seguida, com padrões semelhantes entre masculino e feminino. Embora o volume absoluto seja muito superior para o gênero masculino, a Tabela 6 mostra que

Figura 7 – Primeira classificação de tom - Gênero masculino



Fonte: Gráfico elaborado pela autora (2025)

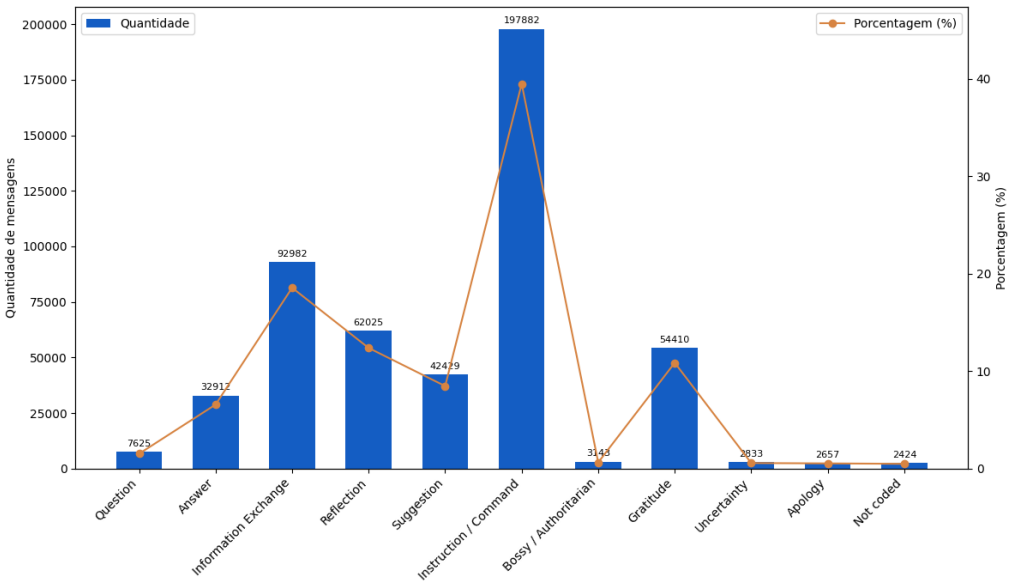
Figura 8 – Segunda classificação de tom - Gênero feminino



Fonte: Gráfico elaborado pela autora (2025)

há uma similaridade nas porcentagens entre os dois gráficos, indicando que, nesta etapa, não foi possível identificar tendências marcantes do modelo em diferenciar o tom da comunicação por gênero.

Figura 9 – Segunda classificação de tom - Gênero masculino



Fonte: Gráfico elaborado pela autora (2025)

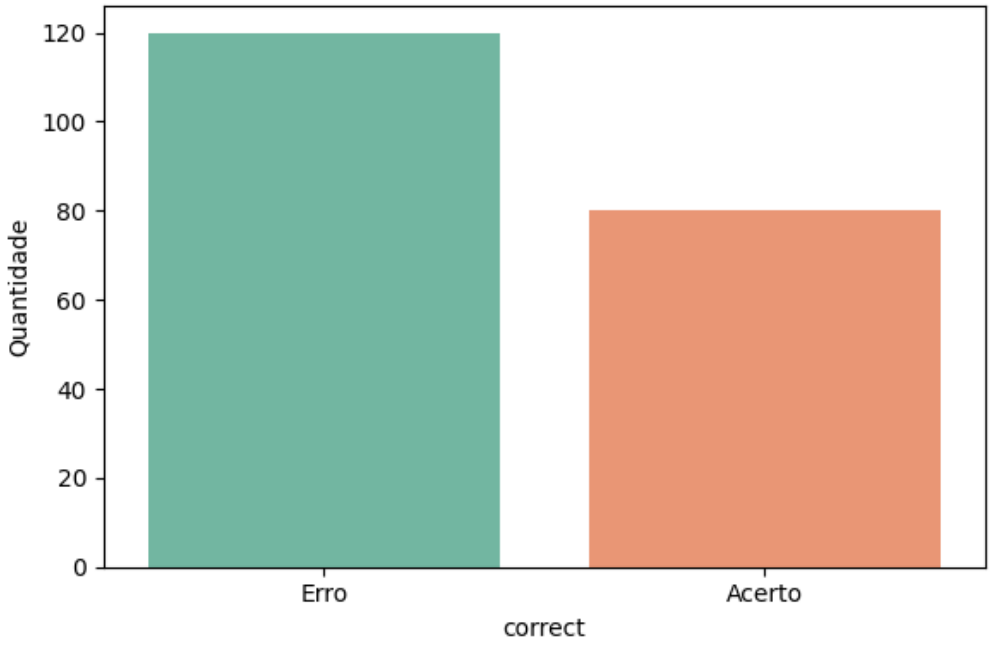
Tabela 6 – Comparação percentual da segunda classificação de tom por gênero

Categoria	**Feminino (%)**	**Masculino (%)**
Question	1.3	1.5
Answer	6.5	6.6
Information Exchange	26.2	18.5
Reflection	12.8	12.4
Suggestion	7.7	8.5
Instruction / Command	37	39.5
Bossy / Authoritarian	0.5	0.5
Gratitude	6.8	10.9
Uncertainty	0.6	0.6
Apology	0.5	0.5
Not Coded	0.2	0.5

A validação da segunda classificação de tom, que foi realizada por meio da anotação manual de uma amostra aleatória de 200 mensagens, resultou em uma acurácia de 40%, classificando corretamente apenas 80 delas (ver Figura 10). Esse resultado indica que, apesar de o modelo ser capaz de identificar corretamente certos padrões linguísticos, ele ainda apresenta um nível significativo de erro na categorização de diferentes tons de comunicação.

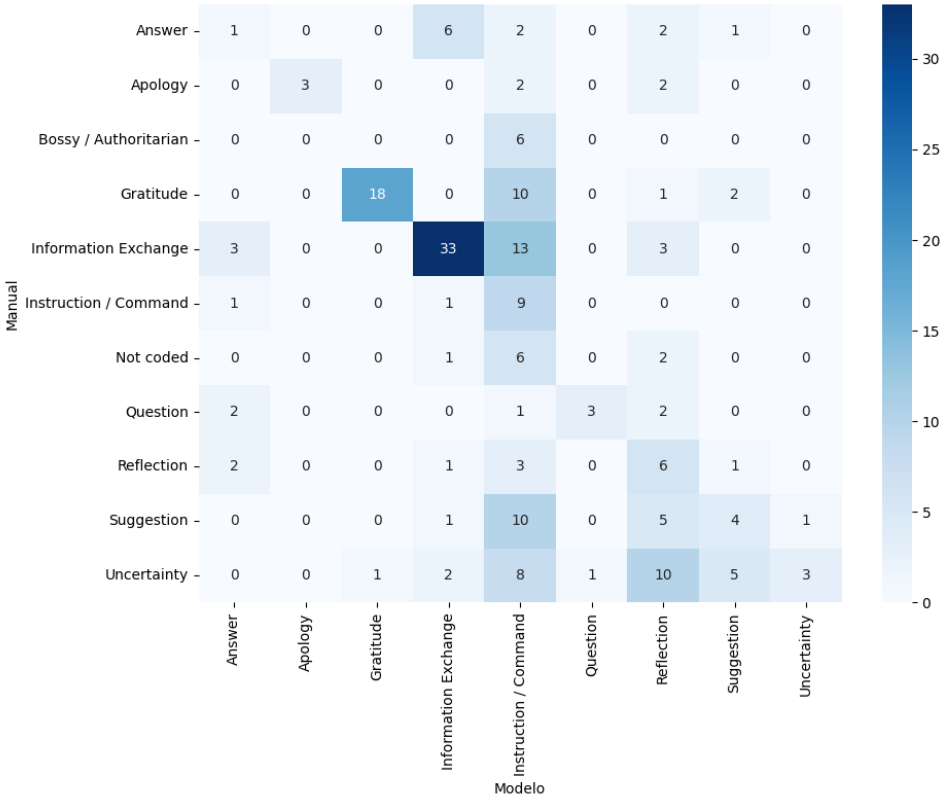
A matriz de confusão, apresentada na Figura 11, mostra a distribuição detalhada entre as classes reais e as preditas pelo modelo. É possível perceber que *"Information Exchange"*

Figura 10 – Acertos e erros do modelo no teste de acurácia



Fonte: Gráfico elaborado pela autora (2025)

Figura 11 – Matriz de confusão do teste de acurácia



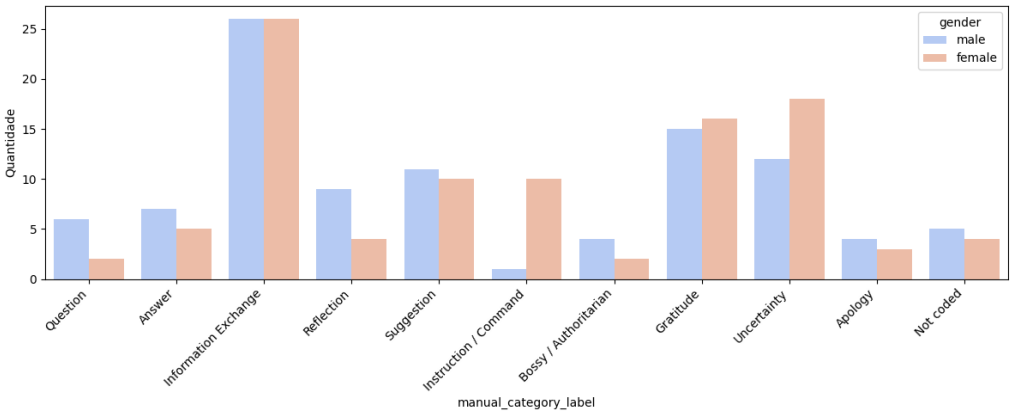
Fonte: Gráfico elaborado pela autora (2025)

apresentou o maior número de classificações corretas, com 33 acertos, seguida de *"Gratitude"*, com 18 acertos. No entanto, há confusões frequentes entre *Information Exchange* e *Instruction/Command* com 13 casos, assim como entre *Gratitude* e *Instruction/Command* que possui 10 casos de erros.

Categorias menos representadas, como *Bossy/Authoritarian* e *Apology*, obtiveram alguns acertos, mas também possuem classificações incorretas. Esse padrão mostra que o modelo costuma confundir tons que têm papéis parecidos na conversa, como dar instruções, trocar informações ou expressar gratidão, o que indica que ele tem dificuldade em perceber algumas nuances na maneira como as mensagens são usadas.

Para visualizar melhor essas diferenças no teste de acurácia, a Figura 12 apresenta a distribuição de mensagens por categoria na classificação manual, enquanto a Figura 13 mostra a mesma distribuição segundo o modelo. Essa comparação ressalta as diferenças na quantidade de mensagens que cada categoria recebeu e também indica que o modelo pode estar exagerando ou deixando de lado algumas classes, provavelmente por causa da forma como ele entende certos padrões de linguagem.

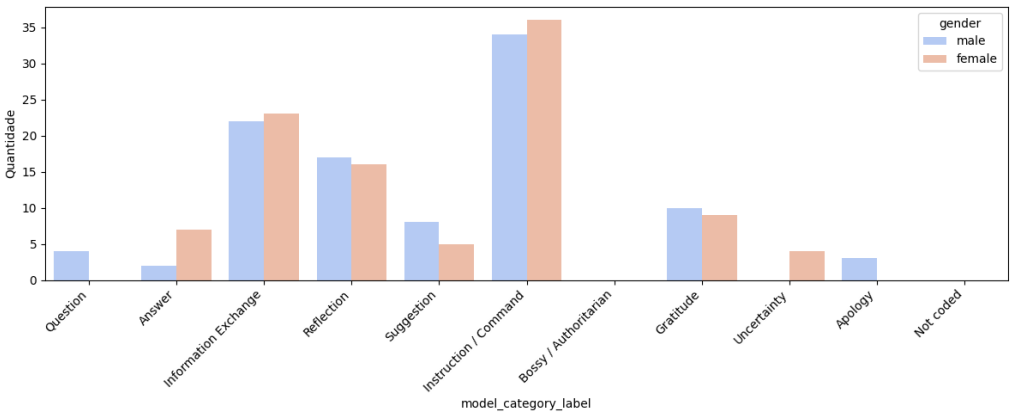
Figura 12 – Distribuição das categorias - Classificação manual



Fonte: Gráfico elaborado pela autora (2025)

Através da Figura 12, observa-se que, apesar de a análise ter sido realizada em uma amostra pequena, os resultados confirmam alguns estudos mencionados anteriormente. Assim como em Newman et al. (2008), o gênero feminino apresentou maior tendência na categoria de incerteza, com 18 mensagens, o que corresponde a 18% contra 12% do gênero masculino. Já em relação ao ponto discutido por Zolduoarrati e Licorish (2021), nota-se que a categoria de reflexão, que representa um tom mais autocentrado, foi predominante entre os homens, enquanto a de instruções, definida no prompt como um comando demonstrado de forma

Figura 13 – Distribuição das categorias - Classificação modelo



Fonte: Gráfico elaborado pela autora (2025)

colaborativa, destacou-se entre as mulheres. Além disso, foi observada uma predominância do gênero masculino na categoria *bossy*, com 4% contra apenas 2% no gênero feminino. Embora a diferença seja pequena, ela reforça o que foi comprovado no estudo sobre a maioria masculina no tom autoritário.

A acurácia baixa (40%) observada na classificação automática do tom das mensagens demonstra as dificuldades presentes na tarefa de interpretação de padrões linguísticos. Mensagens curtas, contextos ambíguos e a diversidade de estilos de comunicação dificultam que o modelo capture com precisão as intenções e emoções presentes. Além disso, há uma subjetividade na definição das categorias de tom e a elas podem se sobrepor entre si, podendo gerar confusão tanto para modelos automatizados quanto para avaliadores humanos. Outro fator que pode contribuir para essa baixa acurácia é a falta de *fine-tuning* específico do modelo para o domínio do desenvolvimento de software e as peculiaridades da comunicação no GitHub.

5 CONCLUSÃO

5.1 CONSIDERAÇÕES FINAIS

O presente trabalho teve como objetivo investigar o uso de modelos de linguagem de grande escala para a identificação e análise de vieses de gênero em ambientes de desenvolvimento colaborativo, com foco nas interações presentes em comentários e revisões de *pull requests* da plataforma GitHub.

Para isso, foi realizado um processo de coleta e pré-processamento de dados utilizando a API do GitHub, seguido da inferência de gênero dos usuários a partir do nome próprio utilizando um modelo LLM. Após isso, as mensagens coletadas foram classificadas em categorias de tom, visando identificar possíveis diferenças na forma de comunicação entre gêneros.

Os resultados indicaram uma predominância interações de usuários masculinos, o que confirma estudos anteriores sobre a sub-representação feminina em ambientes de desenvolvimento open-source. A análise do tom das mensagens revelou padrões semelhantes entre os gêneros em todas as categorias, contradizendo o que foi observado em pesquisas anteriores. Entretanto, a acurácia da classificação automática foi baixa, apontando para desafio presentes em LLMs no reconhecimento preciso das variantes da comunicação. A análise manual, por outro lado, evidenciou diferenças relacionadas a trabalhos prévios, como a maior presença feminina na categoria de incerteza e a predominância masculina na categoria de reflexão, enquanto a de instruções apresentou maior frequência entre as mulheres.

Entre as principais contribuições deste estudo, destacam-se a aplicação de LLMs para a análise de vieses de gênero em ferramentas de controle de versão, a elaboração de um processo automatizado para coleta e classificação de interações em *pull requests* e a validação dos resultados por meio de análise manual, que permitiu uma reflexão sobre as capacidades e limitações do Llama 3.

Em síntese, este estudo contribuiu para o avanço do entendimento sobre viéses de gênero em ambientes de desenvolvimento colaborativo, demonstrando o potencial e os desafios do uso de LLMs como ferramenta para análise automática e ampliando o debate sobre inclusão e diversidade na engenharia de software.

5.2 LIMITAÇÕES

A análise foi conduzida exclusivamente em repositórios públicos e de desenvolvimento open-source do GitHub. Esse ecossistema possui sua própria dinâmica de colaboração, que não necessariamente se reflete em ambientes corporativos ou em projetos privados. Dessa forma, os resultados apresentados não podem ser generalizados para contextos institucionais distintos, como empresas de tecnologia de grande porte, onde fatores como cultura interna e políticas de diversidade podem afetar de maneira diferente as interações.

Apesar de ter sido realizada uma filtragem inicial nos dados para identificar e excluir contas associadas a bots ou usuários automatizados, é possível que alguns desses perfis ainda estejam presentes na base analisada. Os mesmos podem influenciar métricas de comunicação e revisão de forma enviesada, principalmente se gerarem comentários ou aprovações automáticas. Isso pode afetar, por exemplo, a percepção do tom das mensagens e influenciar num quantitativo maior da categoria de aprovação.

Outro fator é que a classificação de tom foi realizada com base em um único modelo LLM, o LLaMA 3. Embora esse modelo apresente desempenho competitivo em tarefas de geração e interpretação de linguagem natural, a aplicação de diferentes LLMs seria necessária para um resultado mais preciso. Como cada arquitetura tem particularidades em seu treinamento, ao fazer a análise utilizando o mesmo prompt e conjunto de dados os resultados podem divergir. Dessa forma, os resultados obtidos podem ser sensíveis à escolha do modelo utilizado.

É importante considerar também que, embora os LLMs apresentem avanços no campo de processamento de linguagem natural, eles podem refletir vieses presentes nos dados com os quais foram treinados. Esses modelos são treinados em grandes volumes de textos coletados da internet, que já incluem padrões culturais, sociais e linguísticos que podem carregar preconceitos implícitos, incluindo vieses de gênero. Por conta disso, a classificação do tom das mensagens pode ser influenciada por esses vieses, afetando a interpretação dos resultados da pesquisa. Por exemplo, o modelo pode ter uma tendência a associar certas expressões a gêneros específicos, mesmo que, no contexto da realidade, essas associações não sejam corretas. Essa limitação reforça a necessidade de uma complementação de uma análise automatizada com avaliações humanas ou outras abordagens para eliminar possíveis vieses algorítmicos.

Por último, a classificação de gênero dos usuários foi realizada com base em seus nomes próprios, utilizando o LLaMA3, e essa abordagem apresenta limitações quanto à precisão. Em particular, a definição de gênero baseada apenas no nome não reflete necessariamente a identi-

dade de gênero real do usuário. Resultados mais precisos seriam obtidos se os dados incluíssem autodeclarações de gênero ou se fosse utilizada uma ferramenta especializada nesse tipo de inferência, como o serviço Genderize.io, que oferece classificações probabilísticas baseadas em grandes bases de dados internacionais.

5.3 TRABALHOS FUTUROS

Como trabalhos futuros, sugere-se explorar outras informações coletadas para investigar possíveis vieses de gênero. Dados como o status dos pull requests, diferenças entre os gêneros na aprovação ou rejeição das contribuições, e o tempo de revisão e aceitação podem fornecer insights importantes para a compreensão e redução desses vieses.

Além disso, recomenda-se a aplicação dessa análise utilizando diferentes LLMs, a fim de comparar e validar os resultados obtidos, contribuindo para a acurácia das conclusões.

Futuras melhorias também poderiam incluir o uso de conjuntos de dados classificados manualmente para permitir um treinamento supervisionado mais preciso e o desenvolvimento de categorias de classificação mais exclusivas. Por fim, o aprofundamento da análise por meio de métodos qualitativos pode ajudar a compreender melhor os contextos específicos das interações, enquanto a integração de fontes de dados adicionais poderia possibilitar uma classificação de gênero mais precisa e inclusiva.

REFERÊNCIAS

AHMED, T.; SRIVASTAVA, A. Understanding and evaluating the behavior of technical users: A study of developer interaction at stackoverflow. *Human-centric Computing and Information Sciences*, v. 7, n. 1, p. 8, 2017. Disponível em: <https://hcis-journal.springeropen.com/articles/10.1186/s13673-017-0091-8>.

BOMMASANI, R.; HUDSON, D. A.; ADELI, E.; ALTMAN, R.; ARORA, S.; ARX, S. V.; BERNSTEIN, M. S.; BOHG, J.; BOSSELUT, A.; BRUNSKILL, E.; BRYNJOLFSSON, E.; BUCH, S.; CARD, D.; CASTELLON, R.; CHATTERJI, N.; CHEN, A.; CREEL, K.; DAVIS, J. Q.; DEMSZKY, D.; DONAHUE, C.; DOUMBOUYA, M.; DURMUS, E.; ERMON, S.; ETCHEMENDY, J.; ETHAYARAJH, K.; FEI-FEI, L.; FINN, C.; GALE, T.; GILLESPIE, L.; GOEL, K.; GOODMAN, N.; GROSSMAN, S.; GUHA, N.; HASHIMOTO, T.; HENDERSON, P.; HEWITT, J.; HO, D. E.; HONG, J.; HSU, K.; HUANG, J.; ICARD, T.; JAIN, S.; JURAFSKY, D.; KALLURI, P.; KARAMCHETI, S.; KEELING, G.; KHANI, F.; KHATTAB, O.; KOH, P. W.; KRASS, M.; KRISHNA, R.; KUDITIPUDI, R.; KUMAR, A.; LADHAK, F.; LEE, M.; LEE, T.; LESKOVEC, J.; LEVENT, I.; LI, X. L.; LI, X.; MA, T.; MALIK, A.; MANNING, C. D.; MIRCHANDANI, S.; MITCHELL, E.; MUNYIKWA, Z.; NAIR, S.; NARAYAN, A.; NARAYANAN, D.; NEWMAN, B.; NIE, A.; NIEBLES, J. C.; NILFOROSHAN, H.; NYARKO, J.; OGUT, G.; ORR, L.; PAPADIMITRIOU, I.; PARK, J. S.; PIECH, C.; PORTELANCE, E.; POTTS, C.; RAGHUNATHAN, A.; REICH, R.; REN, H.; RONG, F.; ROOHANI, Y.; RUIZ, C.; RYAN, J.; Ré, C.; SADIGH, D.; SAGAWA, S.; SANTHANAM, K.; SHIH, A.; SRINIVASAN, K.; TAMKIN, A.; TAORI, R.; THOMAS, A. W.; TRAMèR, F.; WANG, R. E.; WANG, W.; WU, B.; WU, J.; WU, Y.; XIE, S. M.; YASUNAGA, M.; YOU, J.; ZAHARIA, M.; ZHANG, M.; ZHANG, T.; ZHANG, X.; ZHANG, Y.; ZHENG, L.; ZHOU, K.; LIANG, P. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021. Disponível em: <https://arxiv.org/abs/2108.07258>.

CAVALCANTE, L. M. M. *A study on the existence of tolerance to certain code smells and of implicit gender bias in Code Reviews*. [S.l.], 2020. Disponível em: <https://www.cin.ufpe.br/~tg/2020-3/TG_CC/tg_lmmc2.pdf>.

CIN, C. d. i. *Cluster Apuana*. s.d. Disponível em: <https://helpdesk.cin.ufpe.br/servicos/cluster-apuana>. Acesso em: 10 ago. 2025.

Genderize.io. *Genderize API*. 2025. Acesso em: 05 ago. 2025. Disponível em: <https://genderize.io/>.

Genderize.io. *Genderize API Documentation*. 2025. Acesso em: 05 ago. 2025. Disponível em: <https://genderize.io/documentation>.

GitHub. *GitHub*. 2025. <https://github.com/>. Acesso em: Acesso em: 9 ago. 2025.

GITHUB. *Sobre o GitHub e o Git*. 2025. Disponível em: <https://docs.github.com/pt/get-started/start-your-journey/about-github-and-git>. Acesso em: 06 jul. 2025.

Hugging Face. *Hugging Face*. 2025. <https://huggingface.co/>. Acesso em: Acesso em: 9 ago. 2025.

Hugging Face. *Hugging Face*. 2025. <https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct>. Acesso em: Acesso em: 7 jul. 2025.

H.URNA, H. *Pull Request Workflow with Git*. 2020. Disponível em: <https://medium.com/@urna.hybesis/pull-request-workflow-with-git-6-steps-guide-3858e30b5fa4>. Acesso em: 08 ago. 2025.

MAY, A.; WACHS, J.; HANNAK, A. Gender differences in participation and reward on stack overflow. *Empirical Software Engineering*, v. 24, p. 1997–2019, 2019. Disponível em: <https://link.springer.com/article/10.1007/s10664-019-09685-x>.

META. *Introducing Meta Llama 3: The most capable openly available LLM to date*. 2024. Disponível em: <https://ai.meta.com/blog/meta-llama-3/>. Acesso em: 06 jul. 2025.

NEWMAN, M. L.; GROOM, C. J.; HANDELMAN, L. D.; PENNEBAKER, J. W. Gender differences in language use: An analysis of 14,000 text samples. *Discourse Processes*, Taylor & Francis, v. 45, n. 3, p. 211–236, 2008.

O'GRADY, S. *The RedMonk Programming Language Rankings: January 2024*. 2024. Disponível em: <https://redmonk.com/sogrady/2024/03/08/language-rankings-1-24/>. Acesso em: 18 mar. 2025.

OLLAMA. *Chat build with open models*. 2025. Disponível em: <https://ollama.com/>. Acesso em: 07 ago. 2025.

PEREIRA, R.; CATAPAN, E.; SANTOS, N. dos; AL. et (Ed.). *Perspectivas em Engenharia, Mídias e Gestão do Conhecimento – Volume 5*. Editora Arquétipos, 2024. Disponível em: <https://www.researchgate.net/publication/383370521_PERSPECTIVAS_EM_ENGENHARIA_MIDIAS_E_GESTAO_DO_CONHECIMENTO_-_Volume_5>.

PISCITELLI, A. *Gênero: a história de um conceito*. 2009. <http://www.mom.arq.ufmg.br/mom/leiturasfeministas/1a_aula/gÃłnero%20a%20histÃşria%20de%20um%20conceito.pdf>. Acesso em: 9 ago. 2025.

QIU, H. S.; ZHAO, Z. H.; YU, T. K.; WANG, J.; MA, A.; FANG, H.; DABBISH, L.; VASILESCU, B. Gender representation among contributors to open-source infrastructure : An analysis of 20 package manager ecosystems. In: *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*. [S.l.: s.n.], 2023. p. 180–187.

Stack Overflow. *Stack Overflow*. 2025. <https://stackoverflow.com/>. Acesso em: Acesso em: 9 ago. 2025.

SULTANA, S.; TURZO, A. K.; BOSU, A. *Code Reviews in Open Source Projects: How Do Gender Biases Affect Participation and Outcomes?* 2022. Disponível em: <https://arxiv.org/abs/2210.00139>.

SÁNCHEZ, K. R.; SERRANO, G. V.; GÓMEZ, J. P. A.; DURÁN-HERAS, A. Uncovering suggestions in mooc discussion forums: a transformer-based approach. *Artificial Intelligence Review*, v. 58, n. 4, p. 1–23, 2024. Disponível em: <https://doi.org/10.1007/s10462-024-10997-8>.

TERRELL, J.; KOFINK, A.; MIDDLETON, J.; RAINEAR, C.; MURPHY-HILL, E.; PARNIN, C.; STALLINGS, J. Gender differences and bias in open source: pull request acceptance of women versus men. *PeerJ Computer Science*, v. 3, p. e111, 2017. Disponível em: <https://doi.org/10.7717/peerj-cs.111>.

VASILESCU, B.; CAPILUPPI, A.; SEREBRENIK, A. Gender, representation and online participation: A quantitative study. *Interacting with Computers*, p. 488–511, 2013. Disponível em: <https://academic.oup.com/iwc/article/26/5/488/826387>.

VASILESCU, B.; SEREBRENIK, A.; BRAND, M. G. J. van den; POSNETT, D.; RAY, B.; FILKOV, V. Gender and tenure diversity in github teams. In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, 2015. (CHI '15), p. 3789–3798. ISBN 978-1-4503-3145-6. Disponível em: <https://doi.org/10.1145/2702123.2702549>.

WANG, Y.; REDMILES, D. Implicit gender biases in professional software development: An empirical study. In: *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*. [S.l.: s.n.], 2019. p. 1–10.

ZOLDUOARRATI, E.; LICORISH, S. A. On the value of encouraging gender tolerance and inclusiveness in software engineering communities. *Information and Software Technology*, v. 139, p. 106667, 2021. Disponível em: <https://doi.org/10.1016/j.infsof.2021.106667>.

APÊNDICE A – LISTA DE REPOSITÓRIOS COLETADOS

- **Javascript**
  - facebook/react
  - trekhleb/javascript-algorithms
  - airbnb/javascript
  - vercel/next.js
  - f/awesome-chatgpt-prompts
  - Chalarangelo/30-seconds-of-code
  - nodejs/node
  - axios/axios
  - mrdoob/three.js
  - facebook/create-react-app
- **Python**
  - EbookFoundation/free-programming-books
  - public-apis/public-apis
  - donnemartin/system-design-primer
  - vinta/awesome-python
  - TheAlgorithms/Python
  - Significant-Gravitas/AutoGPT
  - AUTOMATIC1111/stable-diffusion-webui
  - huggingface/transformers
  - ytdl-org/youtube-dl
  - 521xueweihan/HelloGitHub
- **Java**
  - Snailclimb/JavaGuide

  - krahets/hello-algo
  - GrowingGit/GitHub-Chinese-Top-Charts
  - iluwatar/java-design-patterns
  - macrozheng/mall
  - doocs/advanced-java
  - spring-projects/spring-boot
  - MisterBooo/LeetCodeAnimation
  - elastic/elasticsearch
  - kdn251/interviews

- **PHP**
  - danielmiessler/SecLists
  - coollabsio/coolify
  - laravel/framework
  - blueimp/jQuery-File-Upload
  - symfony/symfony
  - nextcloud/server
  - composer/composer
  - fzaninotto/Faker
  - filamentphp/filament
  - guzzle/guzzle

- **C#**
  - torvalds/linux
  - Genymobile/scrcpy
  - netdata/netdata
  - redis/redis
  - ventoy/Ventoy

  - obsproject/obs-studio
  - git/git
  - FFmpeg/FFmpeg
  - php/php-src
  - wg/wrk

- **TypeScript**
  - freeCodeCamp/freeCodeCamp
  - kamranahmedse/developer-roadmap
  - vuejs/vue
  - microsoft/vscode
  - yangshun/tech-interview-handbook
  - n8n-io/n8n
  - microsoft/TypeScript
  - langgenius/dify
  - excalidraw/excalidraw
  - angular/angular

- **CSS**
  - animate-css/animate.css
  - ryanoasis/nerd-fonts
  - necolas/normalize.css
  - jgthms/bulma
  - isocpp/CppCoreGuidelines
  - bradtraversy/50projects50days
  - FreeCodeCampChina/freecodecamp.cn
  - houshanren/hangzhou_house_knowledge
  - hehonghui/awesome-english-ebooks

  - Vonng/ddia

- **C++**
  - torvalds/linux
  - Genymobile/scrcpy
  - netdata/netdata
  - redis/redis
  - ventoy/Ventoy
  - obsproject/obs-studio
  - git/git
  - FFmpeg/FFmpeg
  - php/php-src
  - wg/wrk

- **Ruby**
  - rails/rails
  - jekyll/jekyll
  - mastodon/mastodon
  - huginn/huginn
  - maybe-finance/maybe
  - discourse/discourse
  - Homebrew/brew
  - fastlane/fastlane
  - freeCodeCamp/devdocs
  - rapid7/metasploit-framework

- **C**
  - torvalds/linux
  - Genymobile/scrcpy

- netdata/netdata
- redis/redis
- ventoy/Ventoy
- obsproject/obs-studio
- git/git
- FFmpeg/FFmpeg
- php/php-src
- wg/wrk

APÊNDICE B – SCRIPT RESUMIDO PARA COLETA DE PULL REQUESTS NO GITHUB

```python
async def collect_grouped_by_repo():
    headers, out_dir = load_env_and_setup()
    async with aiohttp.ClientSession() as session:
        for lang in ["PHP","C#","TypeScript","CSS","C++","Ruby","C"]:
            repos = await get_top_repos(lang, 10, session, headers)
            for repo in repos:
                full = repo['full_name']
                created = datetime.strptime(repo['created_at'], "%Y-%m-%dT%H:%M:%SZ")
                since = max(created, datetime.utcnow() - timedelta(days=5*365))
                until = datetime.utcnow()

                prs = await get_prs(repo, since, until, session, headers)
                repo_prs = []
                for pr in prs:
                    author_username = pr['user']['login']
                    author_name = await get_user_name(author_username, session, headers)

                    entry = {
                        'repo': full,
                        'pr_id': pr['number'],
                        'pr_title': pr['title'],
                        'pr_author_name': author_name,
                        'pr_author_username': author_username,
                        'merged': pr.get('merged_at') is not None,
                        'state': pr.get('state'),
                        'comments': await get_comments(repo, pr['number'], session, headers),
                        'reviews': await get_reviews(repo, pr['number'], session, headers)
                    }
                    repo_prs.append(entry)
                save_repo_data(repo_prs, full, out_dir)

if __name__ == '__main__':
    asyncio.run(collect_grouped_by_repo())
```

APÊNDICE C – SCRIPT PARA INFERÊNCIA DE GÊNERO

```
def __init__(self, ollama_url="http://localhost:11434", model="llama3"):
    self.ollama_url = ollama_url
    self.model = model
    self.session = requests.Session()

def _create_prompt(self, name: str) -> str:
    return f"""Voce e um especialista em analise de nomes e deve classificar o
        genero do nome fornecido.

    NOME: {name}

    Analise este nome considerando:
    1. Origem cultural e linguistica
    2. Terminacos tipicas por genero
    3. Frequencia de uso historica
    4. Variacoes regionais

    Responda APENAS no formato JSON:
    {"gender": "male" ou "female"}

    Exemplos:
    - Marcos: {"gender": "male"}
    - Kim: {"gender": "female"}
    - Ngozi: {"gender": "female"}

    Classifique o nome: {name}"""

def classify_name(self, name: str) -> Optional[dict]:
    """Classifica um unico nome usando Ollama."""
    prompt = self._create_prompt(name)
    response = self.session.post(
        f"{self.ollama_url}/api/generate",
        json={"model": self.model, "prompt": prompt, "stream": False}
    )
    if response.status_code == 200:
        return json.loads(response.json().get("response", "{}"))
    return None

def load_names(filepath: str) -> List[str]:
"""Carrega nomes de um JSON simples."""
with open(filepath, "r", encoding="utf-8") as f:
    return json.load(f)

def save_results(results: List[dict], filepath: str):
```

```
    """Salva resultados em JSON."""
    with open(filepath, "w", encoding="utf-8") as f:
        json.dump(results, f, indent=2, ensure_ascii=False)

def main():
    names = load_names("names.json")
    classifier = NameGenderClassifier()

    results = []
    for name in names:
        result = classifier.classify_name(name)
        if result:
            results.append({"name": name, **result})

    save_results(results, "names_genderized.json")

if __name__ == "__main__":
    main()
```

APÊNDICE D – PROMPT PARA ANÁLISE DE TOM - PRIMEIRA CLASSIFICAÇÃO

```
You are an expert in analyzing GitHub repository communications. Your task is to classify the following message into exactly one of the 14 categories below.

Categories with Examples:
1. Question: Requests information due to a knowledge gap
Example: "Why was this line removed?"

2. Answer: Provides a direct response to a question or problem
Example: "This function is used to sanitize input from the form."

3. Information Exchange: Shares relevant info from other parts of the project or team
Example: "This bug was already fixed in PR #421."

4. Discussion: Expresses thoughts or opinions about the code or design
Example: "This design seems more flexible in the long term."

5. Approval: Indicates explicit approval of the code or pull request
Example: "LGTM" or "Approved."

6. Reflection: Self-assessment or retrospective insight
Example: "In hindsight, the way we handled error logging could be improved."

7. Scaffolding / Suggestion: Advice or encouragement, often collaborative and constructive
Example: "Maybe we should split this into smaller components."

8. Instruction / Command: Directive or request for change, but stated in a collaborative or polite tone
Example: "Please add tests before merging."

9. Bossy / Authoritarian: Directive expressed in a demanding or forceful tone
Example: "Fix this now."

10. Gratitude: Thanks or praise
Example: "Thanks for fixing this!"

11. Uncertainty: Suggestion or comment with hesitant or uncertain tone
Example: "I'm not sure, but you could try this way."

12. Off-task: Irrelevant to the work item
Example: "Going on vacation tomorrow."

```

```
13. Apology: Expressing regret
Example: "Sorry, I pushed the wrong commit."

14. Not coded: Communications that cannot be assigned to any other category
Example: "Lorem ipsum dolor sit amet..."

Message to classify: "{}"

Instructions:
    - Read the message carefully
    - Compare it with the examples provided
    - Consider the intent and tone
    - Choose only ONE category that best fits
    - Respond with ONLY the category number (1-14)

Classification:
```

APÊNDICE E – PROMPT PARA ANÁLISE DE TOM - TESTE 2

```
You are an expert in analyzing GitHub repository communications. Your task is to classify the following message into exactly one of the 13 categories below.

Categories with Examples:
1. Question: Requests information due to a knowledge gap
Example: "Why was this line removed?"

2. Answer: Provides a direct response to a question or problem
Example: "This function is used to sanitize input from the form."

3. Information Exchange: Shares relevant info from other parts of the project or team
Example: "This bug was already fixed in PR #421."

4. Discussion: Expresses thoughts or opinions about the code or design
Example: "This design seems more flexible in the long term."

5. Approval: Indicates explicit approval of the code or pull request
Example: "LGTM" or "Approved."

6. Reflection: Self-assessment or retrospective insight
Example: "In hindsight, the way we handled error logging could be improved."

7. Scaffolding / Suggestion: Advice or encouragement, often collaborative and constructive
Example: "Maybe we should split this into smaller components."

8. Instruction / Command: Directive or request for change, but stated in a collaborative or polite tone
Example: "Please add tests before merging."

9. Bossy / Authoritarian: Directive expressed in a demanding or forceful tone
Example: "Fix this now."

10. Gratitude: Thanks or praise
Example: "Thanks for fixing this!"

11. Uncertainty: Suggestion or comment with hesitant or uncertain tone
Example: "I'm not sure, but you could try this way."

12. Apology: Expressing regret
Example: "Sorry, I pushed the wrong commit."

13. Not coded: Messages that do not provide technical content, context,
```

```
    suggestions, or identifiable tone.
This includes:
    - Sarcastic, vague or insulting comments
    - Empty mentions or "pinging" messages without content
    - One-liners with unclear intent
    - Off-topic jokes, spam, or meaningless text
Examples:
    - "English teacher"
    - "what a dumb commit lmao"
    - "Pinging @team-name"
    - "losers"
    - "Surely Lucas would like this one"

Message to classify: "{}"

Instructions:
    - Read the message carefully
    - Compare it with the examples and definitions
    - Consider the intent and toneof the message
    - Choose the single most appropriate category
    - If the message lacks clear tone, intent, or relevance, classify as 13
    - Respond with ONLY the category number (1-13)

Classification:
```

APÊNDICE F – PROMPT PARA ANÁLISE DE TOM - TESTE 3

```
You are an expert in analyzing GitHub repository communications. Your task is to classify the following message into exactly one of the 12 categories below.

Categories with Examples:
1. Question: Requests information due to a knowledge gap
Example: "Why was this line removed?"

2. Answer: Provides a direct response to a question or problem
Example: "This function is used to sanitize input from the form."

3. Information Exchange: Shares relevant info from other parts of the project or team
Example: "This bug was already fixed in PR #421." or "This PR was merged into the repository by commit edd7982d5124281c284702820b5dc8a285e6207b."

4. Approval: Indicates explicit approval of the code or pull request
Example: "LGTM" or "Approved."

5. Reflection: Self-assessment or retrospective insight
Example: "In hindsight, the way we handled error logging could be improved."

6. Suggestion: Advice or encouragement, often collaborative and constructive
Example: "We should split this into smaller components."

7. Instruction / Command: Directive or request for change, but stated in a collaborative or polite tone
Example: "Please add tests before merging."

8. Bossy / Authoritarian: Directive expressed in a demanding or forceful tone
Example: "Fix this now."

9. Gratitude: Thanks or praise
Example: "Thanks for fixing this!"

10. Uncertainty: Suggestion or comment with hesitant or uncertain tone
Example: "I'm not sure, but you could try this way." or " I wonder if we need to change this."

11. Apology: Expressing regret
Example: "Sorry, I pushed the wrong commit." or "I didn't mean to submit this."

12. Not coded: Messages that do not provide technical content, context, suggestions, or identifiable tone.
This includes:
```

```
    - Sarcastic, vague or insulting comments
    - Empty mentions or "pinging" messages without content
    - One-liners with unclear intent
    - Off-topic jokes, spam, or meaningless text
Examples:
    - "English teacher"
    - "what a dumb commit lmao"
    - "Pinging @team-name"
    - "losers"
    - "Surely Lucas would like this one"

Message to classify: "{}"

Instructions:
    - Read the message carefully
    - Compare it with the examples and definitions
    - Consider the intent and tone of the message
    - Choose the single most appropriate category
    - If the message lacks clear tone, intent, or relevance, classify as 12
    - Respond with ONLY the category number (1-12)

Classification:
```

APÊNDICE G – PROMPT PARA ANÁLISE DE TOM - TESTE 4

```
You are an expert in analyzing GitHub repository communications. Your task is to classify the following message into exactly one of the 12 categories below.

Categories with Examples:

1. Question: Requests information due to a knowledge gap
Example: "Why was this line removed?"

2. Answer: Provides a direct response to a question or problem
Example: "This function is used to sanitize input from the form."

3. Information Exchange: Shares relevant info from other parts of the project or team
Example: "This bug was already fixed in PR #421."
Example: "This PR was merged into the repository by commit edd7982d..."

4. Approval: Indicates explicit approval of the code or pull request
Example: "LGTM"
Example: "Approved."

5. Reflection: Self-assessment or retrospective insight
Example: "In hindsight, the way we handled error logging could be improved."

6. Suggestion: Advice or encouragement, often collaborative and constructive
Example: "We should split this into smaller components."
Example: "Maybe this could be refactored into a helper function."

7. Instruction / Command: Directive or request for change, but stated in a collaborative or polite tone
Example: "Please add tests before merging."
Example: "You might want to rename this for clarity."

8. Bossy / Authoritarian: Directive expressed in a demanding or forceful tone
Example: "Fix this now."
Example: "This has to be removed."

9. Gratitude: Thanks or praise
Example: "Thanks for fixing this!"
Example: "I thank you!"
Example: "@janober Many thanks!"

10. Uncertainty: Suggestion or comment with hesitant or uncertain tone
Example: "I'm not sure, but you could try this way."
Example: "I wonder if we need to change this."
```

```
42
   11. Apology: Expressing regret
44 Example: "Sorry, I pushed the wrong commit."
   Example: "I didn't mean to submit this."
46
   12. Not coded: Messages that do not provide technical content, context,
       suggestions, or identifiable tone.
48 This includes:
       - Sarcastic, vague or insulting comments
50     - Empty mentions or "pinging" messages without content
       - One-liners with unclear intent
52     - Off-topic jokes, spam, or meaningless text
   Examples:
54     - "English teacher"
       - "what a dumb commit lmao"
56     - "Pinging @team-name"
       - "losers"
58     - "Surely Lucas would like this one"
       - "Got released with n8n@0.68.2"
60
   Message to classify: "{}"
62
   Instructions:
64     - Read the message carefully
       - Compare it with the examples and definitions
66     - Consider the **intent** and **tone** of the message
       - Choose the **single most appropriate category**
68     - If the message lacks clear tone, intent, or relevance, classify as **12**
       - Respond with **ONLY the category number** (1-12)
70
   Classification:
```

APÊNDICE H – PROMPT PARA ANÁLISE DE TOM - TESTE 5

```
You are an expert in analyzing GitHub repository communications. Your task is to classify the following message into exactly one of the 12 categories below.

Categories:

1. Question

2. Answer

3. Information Exchange

4. Approval

5. Reflection

6. Suggestion

7. Instruction / Command

8. Bossy / Authoritarian

9. Gratitude

10. Uncertainty

11. Apology

12. Not coded

Message to classify: "{}"

Instructions:
- Read the message carefully
- Consider the **intent** and **tone** of the message
- Choose the **single most appropriate category**
- If the message lacks clear tone, intent, or relevance, classify as **12**
- Respond with **ONLY the category number** (1-12)

Classification:
```

APÊNDICE I – PROMPT PARA ANÁLISE DE TOM - SEGUNDA ANÁLISE

```
You are an expert in analyzing GitHub repository communications. Your task is to classify the following message into exactly one of the 12 categories below.

Categories with definitions:

1. Question: Requests information due to a knowledge gap

2. Answer: Provides a direct response to a question or problem

3. Information Exchange: Shares relevant info from other parts of the project or team

4. Approval: Indicates explicit approval of the code or pull request

5. Reflection: Self-assessment or retrospective insight

6. Suggestion: Advice or encouragement, often collaborative and constructive

7. Instruction / Command: Directive or request for change, but stated in a collaborative or polite tone

8. Bossy / Authoritarian: Directive expressed in a demanding or forceful tone

9. Gratitude: Thanks or praise

10. Uncertainty: Suggestion or comment with hesitant or uncertain tone

11. Apology: Expressing regret

12. Not coded: Messages that do not provide technical content, context, suggestions, or identifiable tone.
This includes:
 - Sarcastic, vague or insulting comments
 - Empty mentions or "pinging" messages without content
 - One-liners with unclear intent
 - Off-topic jokes, spam, or meaningless text

Message to classify: "{}"

Instructions:
 - Read the message carefully
 - Compare it with the definitions
 - Consider the **intent** and **tone** of the message
 - Choose the **single most appropriate category**
```

```
- If the message lacks clear tone, intent, or relevance, classify as **12**
- Respond with **ONLY the category number** (1-12)

Classification:
```