COPPE
Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia
UFRJ

DESEMPENHO E ECONOMICIDADE DE MODELOS DE LINGUAGEM PARA CLASSIFICAÇÃO DE TOXICIDADE EM JOGOS

Paulo Roberto Xavier Júnior

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Geraldo Bonorino Xexéo

Rio de Janeiro
Agosto de 2024

DESEMPENHO E ECONOMICIDADE DE MODELOS DE LINGUAGEM PARA CLASSIFICAÇÃO DE TOXICIDADE EM JOGOS

Paulo Roberto Xavier Júnior

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Orientador: Geraldo Bonorino Xexéo

Aprovada por: Prof. Geraldo Bonorino Xexéo
Prof. Filipe Braida do Carmo
Prof. Jano Moreira de Souza

RIO DE JANEIRO, RJ – BRASIL
AGOSTO DE 2024

Xavier Júnior, Paulo Roberto

Desempenho e economicidade de modelos de linguagem para classificação de toxicidade em jogos/Paulo Roberto Xavier Júnior. – Rio de Janeiro: UFRJ/COPPE, 2024.

XVI, 97 p.: il.; 29, 7cm.

Orientador: Geraldo Bonorino Xexéo

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2024.

Referências Bibliográficas: p. 81 – 86.

1. toxicidade. 2. classificação. 3. modelo de linguagem. I. Bonorino Xexéo, Geraldo. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Caia 7 vezes; levante-se oito.
(Nanakorobi Yaoki)

Agradecimentos

Dedico este título a minha mãe, Dejanira, por ter acreditado e se sacrificado por mim, me criando com a força, coração e resiliência para estar aqui hoje. Você é meu grande exemplo.

À minha esposa Carolina, por ter me mostrado o que é o amor que admira e exalta, acreditando no meu potencial nos momentos onde eu já não o via mais. Seu suporte e coração gentil são o principal motivo de eu estar aqui hoje.

Ao meu grande amigo Vitor Machado, que me apoiou inequivocamente em todo o processo, e sempre disse que eu conseguiria. Obrigado por ter me empurrado nos últimos quilômetros dessa maratona.

Um agradecimento ao meu orientador, Geraldo Bonorino Xexéo, pela paciência e por ter sido humano nas horas em que mais me foi necessário, sem isso, essa pesquisa jamais teria acontecido.

Ao meu amigo Filipe Braida, por ter ouvido minhas incessantes reclamações da carreira acadêmica, e me ajudado com conselhos e palavras nas horas de maior dificuldade.

Aos meus amigos – os que caminham comigo no mesmo caminho e os que não o fazem mais – por todos os ensinamentos e palavras de apoio. Sem esse legado, eu não teria chegado até aqui.

Ao universo e todas as forças que o compõe, pelas bençãos e por sempre terem me guiado nos momentos mais escuros e caóticos, me levando a conhecer pessoas incríveis que me inspiram a ser uma versão melhor minha.

Agradecimentos às agências de pesquisa Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq, Amazon Web Services - AWS, CAPES e FAPERJ pelo suporte e a Chamada CNPq/AWS Nº 64/2022 – Acesso às Plataformas de Computação em Nuvem da AWS (*Cloud Credits for Research*) que viabilizaram essa pesquisa.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

DESEMPENHO E ECONOMICIDADE DE MODELOS DE LINGUAGEM PARA CLASSIFICAÇÃO DE TOXICIDADE EM JOGOS

Paulo Roberto Xavier Júnior

Agosto/2024

Orientador: Geraldo Bonorino Xexéo

Programa: Engenharia de Sistemas e Computação

A toxicidade em jogos online é praticamente onipresente atualmente, com aproximadamente 86% dos jogadores adultos tendo experienciado alguma forma de assédio em 2022. Diante desses desafios, 91% dos desenvolvedores de jogos veem a toxicidade como um problema crítico. O surgimento dos Grandes Modelos de Linguagem (LLMs), apresenta uma oportunidade para aprimorar a detecção e a classificação de toxicidade, dado suas sofisticadas capacidades de compreensão da linguagem natural.

Esta pesquisa investiga o uso de LLMs para classificação de toxicidade no contexto de um desenvolvedor de jogos, que enfrenta uma significativa toxicidade entre seus jogadores. Foi definida uma metodologia abrangente para selecionar LLMs adequados, desenvolvendo dezesseis *prompts* para análise em dez LLMs. Análises de desempenho, custo e impacto no negócio identificaram os modelos *Llama-3-8B-Instruct* e *GPT-4o* como os de melhor desempenho. Um modelo proposto de impacto no negócio destacou a sensibilidade da receita a classificações incorretas do modelo, enfatizando a importância das métricas de desempenho. O modelo final selecionado, *Llama-3-8B-Instruct*, foi avaliado para uso prático, mostrando competência em conversas claramente tóxicas, mas dificuldade em contextos mais sutis.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

PERFORMANCE AND ECONOMICS OF LANGUAGE MODELS FOR TOXICITY CLASSIFICATION IN GAMES

Paulo Roberto Xavier Júnior

August/2024

Advisor: Geraldo Bonorino Xexéo

Department: Systems Engineering and Computer Science

The toxicity in online games is nearly ubiquitous today, with approximately 86% of adult gamers experiencing some form of harassment in 2022. Given these challenges, 91% of game developers view toxicity as a critical issue. The emergence of Large Language Models (LLMs) presents an opportunity to enhance the detection and classification of toxicity due to their sophisticated natural language understanding capabilities.

This research investigates the use of LLMs for toxicity classification in the context of a game developer facing significant toxicity among its players. A comprehensive methodology was defined to select suitable LLMs, developing sixteen prompts for analysis across ten LLMs. Performance, cost, and business impact analyses identified the *Llama-3-8B-Instruct* and *GPT-4o* models as top performers. A proposed business impact model highlighted the sensitivity of income to incorrect model classifications, emphasizing the importance of performance metrics. The final selected model, *Llama-3-8B-Instruct*, was evaluated for practical use, showing competence in clearly toxic conversations but difficulty in more nuanced contexts.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

BERT	Bidirectional Encoder Representations from Transformers . . . 9
BPMN	Business Process Model and Notation 25
Chain-of-Thought	CoT 24
CNN	Redes Neurais Convolucionais 9
CONDA	CONtextual Dual-Annotated Dataset 10
FN	Falsos Negativos 35
FP	Falsos Positivos 35
FPS	Fist Person Shooter 1, 8
GPT	Generative Pre-trained Transformer 9
GPUs	Graphics Processing Unit 28
IRACIS	Increase Revenue, Avoid Costs e Improve Service 26
KPI	Key Performance Indicator 27
LLM	Large Language Models 2
LSTM	Long Short-Term Memory Networks 9
LTV	Life Time Value 4, 5, 71, 74, 78
NLP	Natural Language Processing 9
RNN	Redes Neurais Recorrentes 9
ROI	Return on Investment 28
SLM	Small Language Model 2

Capítulo 1

Introdução

1.1 Motivação

A toxicidade é amplamente presente em jogos online, com aproximadamente 86% dos jogadores adultos tendo experienciado alguma forma de assédio em 2022. Com um aumento de cerca de 20% em apenas três anos, conforme visto na figura 1.1, o comportamento tóxico tem se tornado uma parte cultural dentro das comunidades de jogos, sendo até mesmo considerados como comportamentos justificáveis (ADL, 2023; KOWERT e KILMER, 2022).

Pesquisas recentes apontam que 60% dos jogadores admitem terem encerrado sessões de jogos temporária ou permanentemente, e que sete a cada dez jogadores evitam jogos com reputação de serem tóxicos (KOWERT e KILMER, 2022). A toxicidade é também uma ameaça à indústria de jogos, representando um prejuízo de mais de 1 bilhão de dólares anualmente[1]. A diferença no valor médio gasto por jogadores em jogos considerados não-tóxicos foi de 54%, aliado a menor tolerância a toxicidade pelas gerações mais novas, revelando uma tendência a busca de experiências menos tóxicas pelos jogadores (KOWERT e KILMER, 2022).

[1]Valor inferido considerando-se a perda de retenção devido à toxicidade de 16% em jogos de *Fist Person Shooter* (FPS), em uma indústria de 10 bilhões de dólares ano (GDC, 2024)

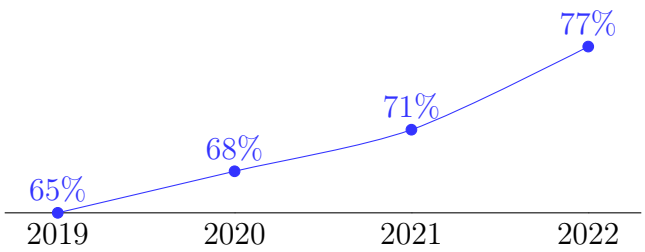


Figura 1.1: Gráfico que mostra a porcentagem de jogadores adultos que experienciaram assédio grave em jogos online entre os anos de 2019 e 2022, onde se pode notar um aumento expressivo nesse tipo de interação (ADL, 2023)

Dentre os desenvolvedores, 91% deles consideram a toxicidade um problema (GDC, 2024). Métodos baseados em regras ou em aprendizado de máquina tradicional podem ter capacidade limitada de detectar nuances na linguagem e de adaptar-se a padrões em constante evolução de comportamento tóxico (SCHMIDT e WIEGAND, 2017). Iniciativas como a *Fairplay Alliance*[2], que hoje conta com mais de 200 desenvolvedoras, busca compartilhar pesquisas e melhores práticas para criar ambientes mais saudáveis para os jogadores e se adaptar à constante evolução nesses sistemas.

Inicialmente, a detecção de toxicidade baseava-se em classificadores clássicos, como o *Naive Bayes*, *Support Vector Machines* (SVM), e *Random Forests*. Embora eficazes em contextos limitados, esses modelos apresentavam limitações significativas em termos de precisão e capacidade de generalização, especialmente quando confrontados com a complexidade e a sutileza da linguagem natural usada em diferentes contextos online. (SCHMIDT e WIEGAND, 2017)

Com os avanços recentes em *Large Language Models* (LLM), como *BERT* (VASWANI *et al.*, 2023) e *GPT* (BROWN *et al.*, 2020), surge uma oportunidade significativa para aprimorar a detecção e a classificação de toxicidade. Esses modelos pré-treinados, capazes de compreender e gerar linguagem natural com um alto grau de sofisticação, oferecem novas possibilidades para enfrentar os desafios inerentes à identificação de comportamentos tóxicos em ambientes dinâmicos e interativos.

Com sua maior capacidade de compreensão de linguagem natural comparado com modelos anteriores, as LLMs se apresentam como uma oportunidade de explorar modelos robustos e precisos para identificar comportamentos tóxicos, através do uso de técnicas como o *fine tuning* para maximizar seu desempenho na tarefa. Devido à correlação entre o número de parâmetros e desempenho do modelo, é possível analisar os resultados e validar o uso das LLMs na tarefa de classificação de toxicidade. No entanto, o uso de LLMs representa um desafio econômico e ambiental, devido a energia e poder computacional necessários para seu uso. (OECD, 2023; NAVEED *et al.*, 2024)

As melhorias em desempenho e capacidade de resolver problemas genéricos trazidas por esses modelos trazem um custo de treinamento[3] e inferência elevado, *e.g.* o custo operacional diário da OpenAI era de aproximadamente 700 mil dólares. Modelos menores tendem a ser mais economicamente viáveis a nível de hardware, e há uma busca contínua por modelos mais eficientes, cunhados como *Small Language Model* (SLM), demonstrando uma preocupação inerente a economicidade dos modelos atuais (NAVEED *et al.*, 2024; LEPAGNOL *et al.*, 2024).

[2] `https://fairplayalliance.org/`

[3] O treinamento do *GPT-4* custou aproximadamente 100 milhões de dólares segundo Sam Altman, CEO da OpenAI

Existem diferenças significativas entre a desempenho dos modelos em diferentes tarefas, de acordo com os dados seu pré-treinamento, arquitetura utilizada, base de dados, dentre outros. O uso de modelos mais robustos não indica necessariamente melhor desempenho em todas as tarefas, onde modelos como o *GPT-3.5*, com 175 bilhões de parâmetros, tem desempenho inferior ao *Llama-3-8B*, com 8 bilhões de parâmetros, em alguns *benchmarks*. (BROWN *et al.*, 2020; TOUVRON *et al.*, 2023)

1.2 Contextualização

A presente pesquisa é conduzida no contexto de uma desenvolvedora de jogos que enfrenta um problema significativo de toxicidade em sua base de jogadores. A toxicidade, se refere a uma ampla variedade de comportamentos negativos e danosos, incluindo assédio ou comunicação abusiva, afetando negativamente a experiência dos jogadores, levando à diminuição da retenção de usuários e potencialmente danificando a reputação da empresa (FROMMEL e MANDRYK, 2022).

Buscando combater a toxicidade, a desenvolvedora inicia a utilização de diferentes tipos de estratégias para mitigação de toxicidade: as de moderação e as que favorecem a criação de uma comunidade de jogo resiliente. Segundo KOWERT e KILMER (2022), uma comunidade de jogos resiliente é aquela que consegue identificar e rejeitar adequadamente comportamentos tóxicos, recuperar-se de perturbações tóxicas, incluindo ataques deliberados, acidentes ou outros incidentes, e resistir a normalização do comportamento tóxico dentro de suas comunidades. Cada estratégia utiliza recursos e elementos diferentes:

- Resiliência da Comunidade de Jogo: utiliza elementos de game design como recompensas e elogios, conforme visto em jogos como *League of Legends* - Riot Games e *Destiny 2* - Bunge, visando aumentar a resiliência da comunidade dos jogos. É uma medida que busca mudar a cultura dos jogadores, garantindo a longevidade da comunidade.

- Moderação: utiliza listas de palavras, detecção automática de toxicidade, uso de moderadores, visando a filtragem e tratamento proativo da toxicidade. É uma medida reativa.

Com um aumento de exposição a toxicidade de aproximadamente 20% em apenas três anos (ADL, 2023), medidas reativas de moderação se tornam necessárias a curto prazo. Devido a sua popularidade e desempenho, a desenvolvedora decide integrar LLMs para melhorar a eficácia de suas ferramentas de detecção de toxicidade, facilitando o trabalho humano de moderação.

Dado que o próprio conceito de tóxico não é trivial, tornando difícil a extração de uma verdade fundamental para estudo (FOUNTA *et al.*, 2018), o racional é que

os LLMs podem oferecer uma compreensão mais profunda do contexto e da sutileza da linguagem, permitindo identificar também situações de abuso mais ambíguas que podem requerer uma análise mais detalhada. Através do *fine-tuning*, podemos treinar os modelos com conjuntos de dados de comunicações de jogadores para aprender a identificar padrões de linguagem e comportamentos que caracterizam a toxicidade, oferecendo desempenho superior (HE *et al.*, 2023).

A utilização deste tipo de modelo traz seus próprios desafios, principalmente economicamente. LLMs são modelos computacionalmente intensivos, que demandam uma grande quantidade de recursos de processamento para treinamento e operação em tempo real (NAVEED *et al.*, 2024), o que incorre em custos que podem ser proibitivos devido a quantidade de dados gerados pelos usuários dos jogos. É necessário que se faça, além de uma investigação de desempenho, uma investigação quanto ao custo e viabilidade financeira desse tipo de solução e seu impacto em termos de negócio, dado que a toxicidade pode afetar a satisfação de um usuário num jogo, essa satisfação pode impactar diretamente sua retenção e, consequentemente, o *Life Time Value* (LTV) daquele usuário.

1.3 Objetivo da Dissertação

O principal objetivo desta dissertação é investigar a eficácia e a viabilidade econômica do uso de Grandes Modelos de Linguagem (LLMs) na classificação de toxicidade em jogos online. Em particular, busca-se avaliar o desempenho desses modelos em identificar comportamentos tóxicos, e analisar os custos associados à sua utilização, a fim de determinar a relação custo-benefício de sua utilização em um ambiente de desenvolvimento de jogos e o impacto da sua escolha pelo negócio.

Para atingir esse objetivo, a pesquisa será guiada pelos seguintes objetivos específicos:

(i) **Proposta de um *framework* para a escolha e utilização de LLMs em tarefas de uso final**: Será proposto um conjunto de etapas estruturadas para selecionar e adaptar LLMs às necessidades específicas do uso de LLMs em tarefas de uso final.

(ii) **Análise do desempenho dos LLMs na classificação de toxicidade:** Testar e comparar diferentes LLMs, em termos de precisão, *recall* e F1-score na identificação de linguagem tóxica.

(iii) **Análise da eficácia de estratégias de *prompting*:** Estratégias como *Chain of Thought*, para melhorar a desempenho desses modelos na tarefa de classificação de toxicidade.

(iv) **Análise do custo de utilização dos LLMs:** Será estimado os custos de inferência dos LLMs e comparar os custos operacionais de diferentes modelos para determinar a viabilidade financeira de sua aplicação em uma desenvolvedora de jogos.

(v) **Proposta de um modelo para o impacto da escolha de LLMs para o negócio:** Será proposto um modelo matemático para correlacionar a satisfação dos usuários com a retenção e o *Life Time Value* (LTV) dos jogadores, avaliando como a detecção de toxicidade afeta esses fatores.

1.4 Resumo dos Resultados

Para responder às perguntas de pesquisa, consideramos a contextualização da seção 1.2. Definimos uma metodologia para escolha de LLMs para tarefas de uso final,a seguindo um conjunto estruturado de etapas. Nessa metodologia, escolhemos a base de dados CONDA (WELD *et al.*, 2021) para uso, e elaboramos um total de dezesseis *prompts* para análise, num total de dez LLMs para analisar o impacto dos *prompts* na tarefa de classificação de toxicidade.

Para escolha do LLM a ser usado, efetuamos uma análise de desempenho, custo e impacto da escolha no negócio. Durante a análise de desempenho, os LLM *Llama-3-8B-Instruct*, *GPT-4o* e *Gemma-7B-Instruct* tiveram os melhores resultados, com a estratégia *Chain of Thought* mostrando-se promissora em grande parte dos modelos. Na análise de custo, relacionamos o custo por inferência com os resultados de desempenho de todos os modelos, selecionando os modelos *Llama-3-8B-Instruct* e *GPT-4o*. Finalmente, analisamos o impacto no negócio da escolha de ambos os modelos.

Para a análise de impacto do negócio, foi proposto um modelo que relaciona principalmente a satisfação do usuário como o pilar na retenção, impactando o *Life Time Value* (LTV) daquele jogador e consequentemente a receita da desenvolvedora. Foi executada uma análise de sensibilidade desse modelo, que mostrou que a variação de receita é particularmente sensível às classificações incorretas do modelo, o que destaca a importância das métricas de desempenho. A relação entre retenção do usuário dada por sua satisfação no jogo e seu potencial LTV também se mostraram relevantes nesse modelo, demonstrando que a satisfação do jogador e sua retenção tem um grande impacto na receita, raciocínio amplamente discutido no decorrer deste trabalho.

Por fim, selecionamos o LLM *Llama-3-8B-Instruct*, provando-se adequado a tarefa. Executamos também uma breve análise de seus resultados, a fim de entender os casos de uso apropriados e onde não desempenha como esperado. O classificador

comporta-se adequadamente em conversações onde a toxicidade é clara, com o uso de palavras consideradas tóxicas, como palavrões. O mesmo tende a errar em conversações onde essas palavras são usadas para autodepreciação, ou fora de contexto, implicando que o LLM classifica toxicidade com muito foco em palavras isoladas, ao invés de utilizar o contexto completo.

1.5 Contribuições

Esta dissertação está contextualizada na área de Aprendizado de Máquina, especificamente no uso de LLMs para classificação de toxicidade, contribuindo nessa área por apresentar: um *framework* para a escolha e utilização de LLMs em uma tarefa de uso final, uma avaliação de desempenho e custo de LLMs em diversos tipos de estratégia de *prompt* e um modelo para impacto no negócio causado pela escolha do LLM.

1.6 Organização deste trabalho

Esta dissertação está organizada da seguinte forma: nos capítulos 2 e 3 serão apresentados os conceitos referentes a fundamentação teórica do trabalho, sobre toxicidade em jogos online e LLMs com arquitetura *Transformer*, respectivamente. No capítulo 4 é apresentada uma metodologia para escolha e utilização de LLMs em uma tarefa de uso final, estruturado em etapas; definimos também um modelo matemático teórico para analisar o impacto da escolha de LLMs no negócio.

No capítulo 5 é apresentado a metodologia utilizada na classificação de toxicidade em jogos online, e os resultados obtidos nas análises de desempenho, custo e impacto no negócio, seguido da escolha do modelo. Finalmente, no capítulo 6 serão apresentadas as considerações finais e principais direções de pesquisa referentes a este trabalho.

Capítulo 2

Toxicidade em jogos

Este capítulo destaca o impacto da toxicidade nos usuários da internet como um todo, seguido do seu impacto social e econômico em jogos online. Discutimos as abordagens atuais de classificação de toxicidade e sobre os trabalhos relacionados relevantes de classificação de toxicidade em jogos online.

2.1 Toxicidade

Usuários de internet percebem diariamente como outros usuários se comportam de forma diferente de como agiriam ordinariamente, se expressando de forma mais aberta e menos contida. Esse efeito universal é conhecido como o efeito da desinibição online e ele pode ocorrer nas duas direções: a desinibição benigna e a desinibição tóxica (SULER, 2004).

A desinibição tóxica, nos leva a presenciar linguagem rude, criticismos, raiva, preconceito e até mesmo ameaças em ambientes online. O termo toxicidade se refere a uma ampla variedade de comportamentos negativos e danosos, incluindo assédio ou comunicação abusiva (FROMMEL e MANDRYK, 2022). A própria definição de conteúdo tóxico não é trivial, o que torna difícil a extração de uma verdade fundamental para estudo (FOUNTA *et al.*, 2018).

Nas mídias sociais, a toxicidade se manifesta de forma ainda mais acentuada devido ao amplo alcance e à rápida disseminação de informações. Estudos têm mostrado que a exposição constante a conteúdo tóxico nas redes sociais está associada a aumentos nos níveis de estresse, ansiedade e depressão entre os usuários (HEMENDINGER, 2023).

Enquanto a toxicidade em mídias sociais é um problema significativo, é importante também examinar como esses comportamentos se manifestam em outras áreas da internet. Um exemplo notório é o ambiente dos jogos online, onde a interatividade e a competitividade exacerbam ainda mais a desinibição tóxica (KOWERT e OLDMEADOW, 2015).

2.2 Toxicidade em Jogos Online

A toxicidade em jogos online acarreta em consequências negativas para a experiência do jogador, provocando angústia psicológica. Essas atitudes também resultam em um decréscimo no desempenho individual e coletivo, um impacto negativo na receita dos desenvolvedores de jogos e na possibilidade de gerar traumas específicos de gênero ou raciais (BERES *et al.*, 2021). Segundo FROMMEL e MANDRYK (2022), há também uma natureza cíclica na toxicidade, levando à normalização da mesma como um elemento aceitável da experiência de jogo.

Pesquisas recentes apontam que 60% dos jogadores admitem terem encerrado sessões de jogos temporária ou permanentemente e que sete a cada dez jogadores evitam jogos devido a sua reputação como tóxico (KOWERT e KILMER, 2022). A toxicidade é também uma ameaça à indústria de jogos, representando um prejuízo de mais de um bilhão de dólares anual[1]. A diferença no valor médio gasto por jogadores em jogos considerados não-tóxicos foi de 54%, e aliado a menor tolerância a toxicidade pelas gerações mais novas, revela uma tendência a busca de experiências menos tóxicas pelos jogadores (KOWERT e KILMER, 2022).

Uma das principais abordagens ao lidar com toxicidade em jogos online, é a classificação de toxicidade (FROMMEL e MANDRYK, 2022). Para lidar com esse desafio, pesquisadores e empresas usam bases de dados rotuladas, e treinam modelos de aprendizado de máquina para detecção, inicialmente com classificadores clássicos, seguindo de modelos de linguagem pré treinados (ZHANG *et al.*, 2018), e agora LLMs, como no caso do *GPT-4* (NAVEED *et al.*, 2024; HE *et al.*, 2023).

O maior obstáculo em desenvolver classificadores de toxicidade é a falta de bases de dados rotuladas que possuam diferentes tipos de conteúdo tóxico (HE *et al.*, 2023). Em plataformas como a *Wikipedia*[2], estima-se que apenas de 1% dos comentários possam ser classificados como tóxicos. Isso revela um dos desafios desse tipo de estudo: encontrar esse conteúdo perante o número massivo de dados disponíveis (ZHANG *et al.*, 2018). Dado o volume de dados e os diversos tipos de toxicidade, uso de *crowdsourcing* provou-se promissor no desenvolvimento dessas bases, conforme demonstrado em estudos como ZHANG *et al.* (2018), BLACKBURN e KWAK (2014) e WELD *et al.* (2021).

Segundo HOWE e JEFF (2006), *crowdsourcing* é a prática de obter ideias, serviços ou conteúdo solicitando contribuições de um grande grupo de pessoas, geralmente de uma comunidade online, para realizar tarefas que tradicionalmente seriam executadas por empregados ou contratados específicos. Essa abordagem permite a rotulagem de grandes volumes de dados, aproveitando a inteligência coletiva e a

[1]Valor inferido considerando-se a perda de retenção devido à toxicidade de 16% em jogos de *Fist Person Shooter* (FPS), em uma indústria de dez bilhões de dólares ano (GDC, 2024)

[2]`https://wikipedia.org`

diversidade de perspectivas dos colaboradores.

Na próxima seção discutiremos a evolução das abordagens para classificação de toxicidade ao decorrer do tempo.

2.3 Trabalhos Relacionados

Inicialmente, a detecção de toxicidade baseava-se em classificadores clássicos como o *Naive Bayes*, *Support Vector Machines* (SVM), e *Random Forests*. Esses métodos utilizavam características textuais simples, como a frequência de palavras e n-gramas, para identificar conteúdos tóxicos. Embora eficazes em contextos limitados, esses modelos apresentavam limitações significativas em termos de precisão e capacidade de generalização, especialmente quando confrontados com a complexidade e a sutileza da linguagem natural usada em diferentes contextos online (SCHMIDT e WIEGAND, 2017).

Com o surgimento do *deep learning*, a detecção de toxicidade avançou significativamente. Modelos de *Redes Neurais Convolucionais* (CNN) e *Redes Neurais Recorrentes* (RNN), incluindo *Long Short-Term Memory Networks* (LSTM), começaram a ser empregados para capturar as dependências contextuais e sequenciais da linguagem. Esses modelos superaram os classificadores clássicos ao considerar a estrutura das frases e o contexto mais amplo, melhorando a capacidade de identificar nuances na comunicação tóxica (KAZBEKOVA *et al.*, 2023; AGRAWAL e AWEKAR, 2018).

Mais recentemente, a introdução de LLMs pré-treinados, como *Bidirectional Encoder Representations from Transformers* (BERT) e *Generative Pre-trained Transformer* (GPT), possibilitou novas abordagens para a classificação de toxicidade. Esses modelos, treinados em enormes volumes de dados, conseguem entender e gerar texto com um alto nível de sofisticação. Por exemplo, BERT, que utiliza uma arquitetura de transformadores bidirecional, permite que o modelo compreenda o contexto de uma palavra baseada tanto no que vem antes quanto no que vem depois dela em uma sentença (DEVLIN *et al.*, 2019).

O uso de LLMs, como o *GPT-3* e o mais recente *GPT-4o*, têm mostrado resultados promissores em tarefas de uso final, incluindo classificação (NAVEED *et al.*, 2024). Esses modelos conseguem capturar e interpretar melhor as sutilezas da linguagem natural, incluindo sarcasmo, ironia e outros tons implícitos que são frequentemente difíceis de detectar com métodos mais simples (BROWN *et al.*, 2020).

Esses avanços demonstram que a evolução da classificação de toxicidade reflete o progresso das técnicas de *Natural Language Processing* (NLP). Na próxima seção discutiremos especificamente os trabalhos relacionados a classificação de toxicidade em jogos.

2.4 Classificação de Toxicidade

A classificação de toxicidade em jogos online difere significativamente da classificação de toxicidade em conversas normais devido a vários fatores contextuais e dinâmicos inerentes aos ambientes de jogos. Em jogos online, a interação é frequentemente marcada por uma alta competitividade, anonimato e uma comunicação rápida e impulsiva, que podem intensificar comportamentos tóxicos diante de eventos contextuais negativos ao jogador *e.g.* mortes, derrota, perda de um objetivo (MARTENS *et al.*, 2016).

BLACKBURN e KWAK (2014) utilizaram um grande conjunto de dados do Tribunal de *League of Legends*[3] com mais dez milhões de conversas, rotulados através de *crowdsourcing* para propor uma abordagem de aprendizado supervisionado para prever as decisões coletivas, obtendo bom desempenho especialmente em casos majoritariamente definidos como tóxicos pelos jogadores, e demonstrando que técnicas de aprendizado de máquina podem complementar sistemas de *crowdsourcing* de forma a reduzir custos operacionais.

A linguagem usada nos jogos é altamente específica, com gírias e jargões que não são comuns em conversas normais, tornando a detecção de toxicidade mais desafiadora. MARTENS *et al.* (2016); DE MESQUITA NETO e BECKER (2018), analisaram os padrões das conversações de jogos, levando a criação de um *framework* com tópicos utilizados, separando-os entre positivos e negativos, relacionando-o com o desempenho dos grupos de jogadores dado a prevalência de seus padrões conversações, indicando uma perda de desempenho nos jogos em ofensores.

O *framework* desenvolvido por DE MESQUITA NETO e BECKER (2018), foi futuramente expandido por WELD *et al.* (2021), que criou a base de dados *CONtextual Dual-Annotated Dataset* (CONDA), utilizando-se de *crowdsourcing*, com anotações a nível de palavra e de sentença, o que será discutido em maiores detalhes na seção 5.4. Neste trabalho, o desempenho de diferentes algoritmos é comparada usando a base de dados, fazendo uso de algoritmos de RNNs, LSTMs e BERT. O algoritmo com melhor desempenho do grupo foi o *Joint Bert* (CHEN *et al.*, 2019), com um acurácia de 83% aproximadamente.

O modelo *Toxbuster*, proposto por YANG *et al.* (2023), utiliza *BERT* como base para detecção de toxicidade de jogos online, o comparando com outros modelos de detecção de toxicidade, como o *CleanSpeak*[4], *Detoxify*[5] e *PerspectiveAPI*[6], obtendo o melhor resultado com uma precisão de 83% aproximadamente.

[3]Sistema introduzido em maio de 2011 pela *Riot Games Inc.* em *League of Legends* onde jogadores podiam analisar casos de comportamento anti-social e puni-los. Foi retirado do ar e substituído por sistemas automatizados em 2014

[4]`https://cleanspeak.com/docs/3.x/tech/apis/`

[5]`https://github.com/unitaryai/detoxify`

[6]`https://perspectiveapi.com`

Trabalhos como HE *et al.* (2023); KOH *et al.* (2024); DE WYNTER *et al.* (2024) avaliaram o desempenho de LLMs na tarefa de classificação de toxicidade utilizando de *BERT* e modelos mais recentes, como *Llama-2* e *GPT-3*.

No próximo capítulo, discutiremos os LLMs que surgiram como uma evolução dos modelos de linguagem, impulsionados pela arquitetura *Transformer* (VASWANI *et al.*, 2023).

Capítulo 3

Large Language Models com Arquitetura Transformer

Modelos de linguagem são sistemas de inteligência artificial treinados para entender, gerar e manipular texto em linguagem natural. Eles são uma parte fundamental do campo de Processamento de Linguagem Natural, que abrange técnicas e métodos para a análise e síntese de linguagem humana. No contexto de NLP, os modelos de linguagem são utilizados para uma ampla variedade de tarefas, como tradução automática, análise de sentimentos, resumo de textos e resposta a perguntas. Eles operam aprendendo padrões e estruturas a partir de grandes corpos de textos, permitindo-lhes prever a próxima palavra em uma sequência, completar frases ou até mesmo gerar novos textos coerentes e contextualmente apropriados (RADFORD *et al.*, 2019).

Os LLMs surgiram como uma evolução natural dos modelos de linguagem, impulsionada pelo aumento exponencial na capacidade computacional e na disponibilidade de grandes volumes de dados textuais (VASWANI *et al.*, 2023; NAVEED *et al.*, 2024). Antes dos LLMs, os modelos de linguagem eram limitados em termos de escala e capacidade de generalização.

Com a introdução de arquiteturas como o Transformer, tornou-se viável treinar modelos com bilhões de parâmetros, como o GPT da OpenAI e o BERT da Google. Esses LLMs são capazes de capturar nuances complexas da linguagem, realizar inferências mais sofisticadas e fornecer respostas mais precisas e contextualmente relevantes. Desde a sua introdução, esses modelos recebem uma atenção cada vez maior entre o público geral e o acadêmico, como pode ser visto na figura 3.1.

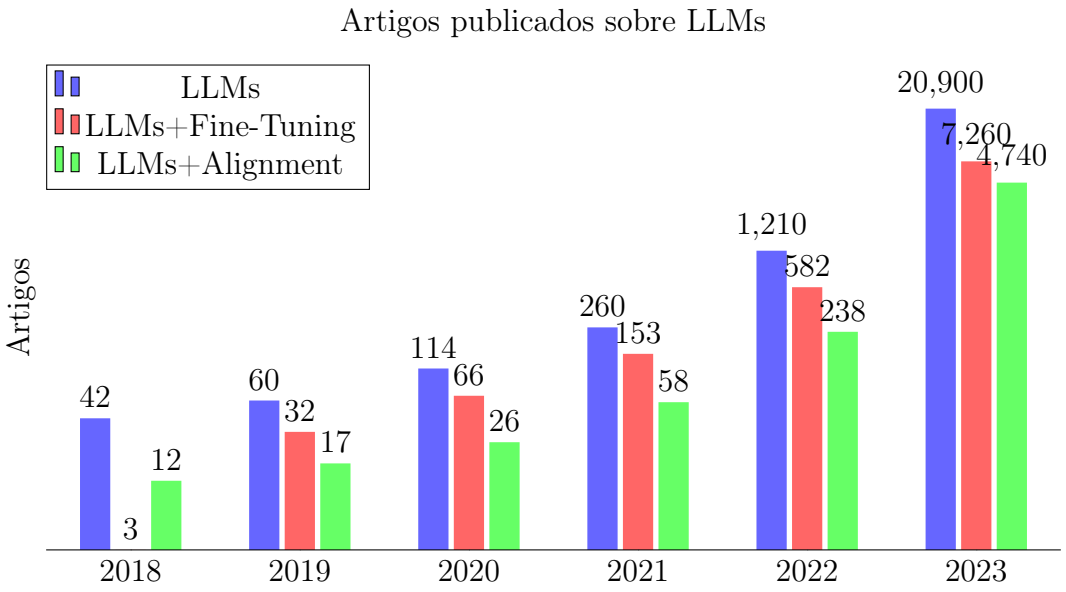


Figura 3.1: As séries mostram a quantidade de publicações ao longo do tempo contendo as palavras-chave "Large Language Model", "Large Language Model + Fine-Tuning" e "Large Language Model + Alignment" (OECD, 2023).

3.1 Large Language Models (LLMs)

Os Large Language Models (LLMs) são utilizados em uma vasta gama de aplicações devido à sua capacidade de compreender e gerar texto com alta precisão fatual e correlação com a entrada (NAVEED *et al.*, 2024). Eles são empregados em assistentes virtuais como *Siri*[1] e *Alexa*[2], na tradução automática em serviços como *Google Translate*, na criação de conteúdo automatizado, na análise de sentimentos para monitoramento de redes sociais e em muitas outras áreas. A capacidade dos LLMs de realizar tarefas de linguagem com um nível de sofisticação próximo ao humano os torna ferramentas poderosas para melhorar a eficiência e a qualidade das interações baseadas em texto.

Os principais modelos LLMs que marcaram a evolução da tecnologia incluem:

- GPT (Generative Pre-trained Transformer): Desenvolvido pela *OpenAI*, o *GPT-3* é um dos mais conhecidos, com 175 bilhões de parâmetros, destacando-se por sua habilidade de gerar texto altamente coerente e relevante (BROWN *et al.*, 2020).

- BERT (Bidirectional Encoder Representations from Transformers): Criado pela *Google*, *BERT* é conhecido por sua abordagem bidirecional ao processa-

[1] `https://www.tomshardware.com/tech-industry/artificial-intelligence/apple-intelligence-siri-gets-an-llm-brain-transplant-chatgpt-integration-and-genmojis`

[2] `https://www.amazon.science/blog/alexa-unveils-new-speech-recognition-text-to-speech-technologies`

mento de texto, permitindo uma melhor compreensão do contexto (DEVLIN *et al.*, 2019).

- T5 (Text-To-Text Transfer Transformer): Outro modelo da *Google* que reformula todas as tarefas de NLP como problemas de tradução de texto para texto, facilitando uma abordagem unificada para diferentes aplicações (RAFFEL *et al.*, 2023).

- XLNet: Desenvolvido pela *Google/CMU*, combina as vantagens do *BERT* e de métodos autoregressivos como o *GPT*, utilizando uma abordagem permutacional para modelagem de linguagem (YANG *et al.*, 2020).

Algo que todos esses modelos tem em comum, é o grande número de parâmetros existentes em suas redes. Como pode ser visto na figura 3.2, existe uma tendência de aumento desses parâmetros, visto que isso tem uma relação direta com a precisão dos modelos na realização de suas tarefas.

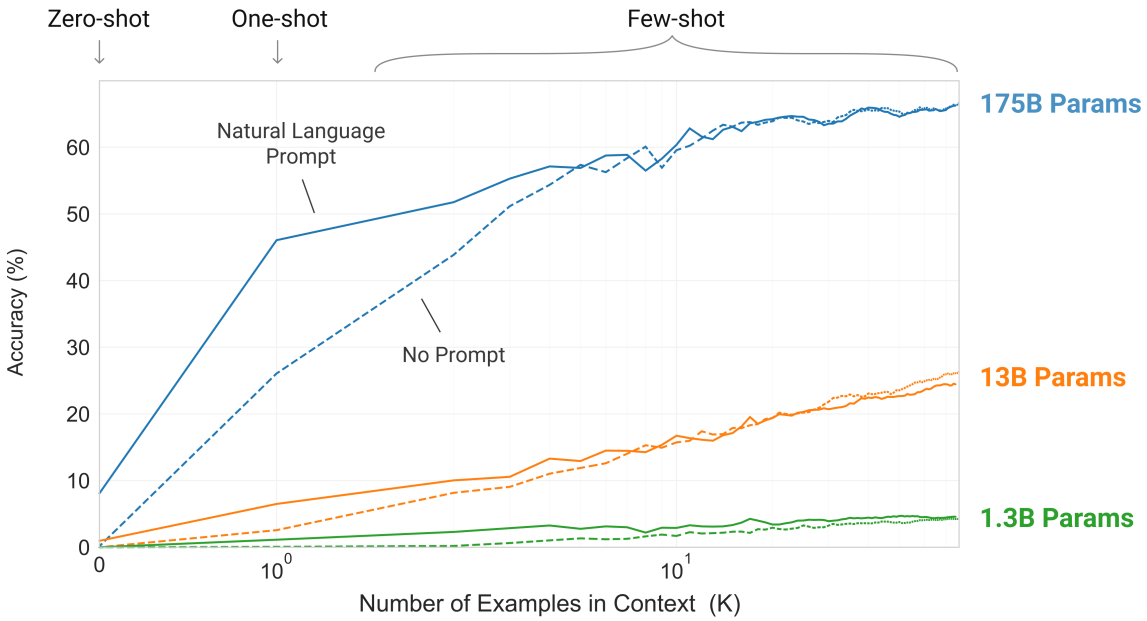


Figura 3.2: Modelos treinados em maior número de parâmetros tendem a ter um melhor desempenho (BROWN *et al.*, 2020)

O foco desse trabalho será nos modelos com arquitetura *Transformer*, devido à sua eficiência em lidar com entradas de texto que possuem contextos extensos, e à sua versatilidade proveniente do uso de *prompts*. Veremos essa arquitetura em mais detalhes na seção 3.3, porém é importante estabelecer o conceito de embeddings, que é descrito brevemente na seção 3.2.

3.2 Embeddings

Embeddings são representações vetoriais densas de dados, que permitem a transformação de entradas discretas e de alta dimensionalidade (como palavras ou frases) em vetores de números reais com menor dimensionalidade. Esses vetores capturam as características semânticas e sintáticas das entradas, o que é muito útil para representação interna dessas entradas dentro de um modelo de linguagem. A proximidade entre vetores reflete similaridade semântica entre as entradas.

Para ilustrar o conceito de embeddings, considere o seguinte exemplo simples. Suponha que temos um corpus de frases curtas e desejamos gerar embeddings para as palavras contidas nele.

```
Corpus:
1. O gato está no tapete.
2. O cachorro está no jardim.
3. O gato e o cachorro são amigos.
```

Usando uma técnica de geração de embeddings como a *Word2Vec* (MIKOLOV *et al.*, 2013), podemos construir um modelo que gerará os seguintes vetores para algumas das palavras presentes no corpus de frases:

```
"gato": [0.12, 0.43, 0.67, 0.89]
"cachorro": [0.15, 0.40, 0.65, 0.88]
"tapete": [0.90, 0.15, 0.35, 0.42]
"jardim": [0.52, 0.33, 0.33, 0.97]
```

Aqui, a proximidade dos vetores entre "gato" e "cachorro" reflete uma similaridade semântica maior em comparação com "tapete" e "jardim".

3.3 Arquitetura Transformer

A arquitetura *Transformer*, introduzida em (VASWANI *et al.*, 2023), é revolucionária porque elimina a dependência de estruturas sequenciais, como Redes Neurais Recorrentes (RNNs), ao empregar um mecanismo denominado "auto-atenção" que pode ser processado de maneira paralela. Isso resulta em uma eficiência computacional significativamente maior e na capacidade de capturar dependências de longo alcance em textos de forma mais eficaz. Os *Transformers* são compostos por camadas de codificadores e decodificadores, onde cada camada aplica operações de auto-atenção e feed-forward, permitindo que o modelo enfoque em diferentes partes da entrada simultaneamente (PRINCE, 2024).

3.3.1 Camadas encoder-decoder

A arquitetura *encoder-decoder* é uma estrutura fundamental em modelos de aprendizado profundo, especialmente usada em tarefas de sequência para sequência ("*seq2seq*"), como tradução automática, resumo de texto e resposta a perguntas. Vamos detalhar essa arquitetura e sua relação com o mecanismo de atenção (PRINCE, 2024).

O *encoder* é responsável por processar a sequência de entrada e codificá-la em uma representação intermediária, que captura a informação essencial da entrada. Consiste em várias camadas, cada uma composta por subcamadas de auto-atenção e *feed-forward*. Toma como entrada uma sequência de *embeddings* das palavras e passa essa sequência através de suas camadas. A saída do *encoder* é uma sequência de vetores de contexto, um para cada posição na sequência de entrada.

O *decoder* gera a sequência de saída, uma palavra de cada vez, utilizando a representação intermediária fornecida pelo *encoder* através dos vetores de contexto. Assim como o *encoder*, o *decoder* consiste em várias camadas, cada uma composta por subcamadas de auto-atenção, atenção cruzada (ou inter-atenção) e feed-forward. O *decoder* recebe os *embeddings* da sequência de saída previamente gerada (ou uma sequência de início durante o treinamento) e a saída do *encoder*. Ele gera a próxima palavra na sequência de saída, um passo de cada vez.

Um esquema simples da arquitetura pode ser visto na figura 3.3. O ponto central do fluxo, o vetor de contexto, é construído com o uso do mecanismo de auto-atenção que será descrito na seção 3.3.2.

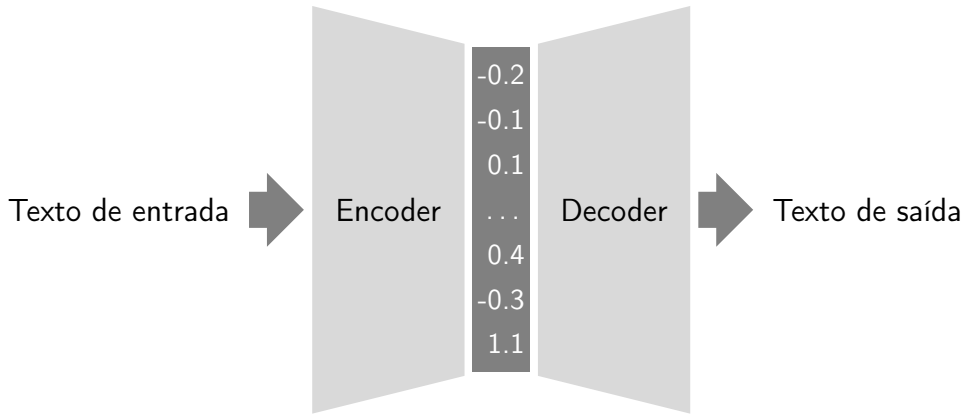


Figura 3.3: Fluxo básico da arquitetura *Encoder-decoder* (PRINCE, 2024)

3.3.2 Mecanismo de auto-atenção

Uma parte vital da arquitetura *Transformer*, e que permite que esses modelos tenham um desempenho muito bom no entendimento do contexto das sentenças, é o mecanismo de auto-atenção. Esse mecanismo é fundamental para os *Transformers*

e permite que o modelo avalie a importância de cada palavra em relação às outras em uma sentença (VASWANI *et al.*, 2023).

O mecanismo de auto-atenção funciona calculando uma pontuação de atenção para cada par de palavras na sequência de entrada. Essas pontuações determinam o quanto cada palavra deve influenciar a representação das outras. Especificamente, o processo envolve três matrizes denominadas $Query$ (Q), Key (K) e $Value$ (V). As pontuações de atenção são obtidas pelo produto escalar entre $Query$ e Key, seguido de uma normalização (função $softmax$), que é então multiplicada pelos $Value$ para produzir as representações ponderadas. Isso permite ao modelo focar nas partes relevantes da entrada, independentemente de sua posição na sequência.

Para explicar os conceitos das matrizes $Query$, Key e $Value$ de maneira intuitiva, vamos fazer uma analogia com uma biblioteca onde você está tentando encontrar informações específicas em livros.

- $Query$ (Consulta): Imagine que você entra na biblioteca e tem uma pergunta específica em mente, por exemplo, "Quais são os principais eventos da Revolução Francesa?". Essa pergunta será a sua "consulta".

- Key (Chave): Dentro da biblioteca, cada livro tem um índice ou um sumário que descreve seu conteúdo. Esses índices ou sumários representam a chave. Cada livro tem uma chave que descreve as informações contidas nele.

- $Value$ (Valor): O conteúdo real dos livros, os textos, figuras e informações que eles contêm, são os seus valores. O valor é o que você realmente quer acessar e ler para obter a resposta à sua pergunta.

Agora, imagine que você está tentando encontrar a resposta para sua pergunta na biblioteca usando os conceitos de consulta, chave e valor.

- Formulação da consulta:

  Você pensa na pergunta "Quais são os principais eventos da Revolução Francesa?" e isso é sua consulta.

- Comparação da consulta com as chaves dos livros:

  Você começa a olhar para os índices ou sumários dos livros (chaves) para ver quais deles mencionam a Revolução Francesa e eventos importantes.

  Essencialmente, você está tentando encontrar os livros cujos índices (chaves) correspondem bem à sua pergunta (consulta).

- Encontrando os livros relevantes e acessa os seus valores:

Quando você encontra um índice (chave) que parece relevante, você decide que vale a pena abrir o livro e ler o conteúdo.

O conteúdo que você lê é o valor.

Aplicando esses conceitos para o contexto do mecanismo de atenção, as matrizes podem ser entendidas como:

- $Query$ (Q): São as representações vetoriais das palavras que você está processando e que estão tentando encontrar informações relevantes em outras palavras da sequência.
- Key (K): São as representações vetoriais das mesmas palavras, mas vistas como "índices" que descrevem quais informações essas palavras possuem.
- $Value$ (V): São também representações vetoriais das palavras, mas representam o conteúdo real ou a informação que cada palavra carrega.

Considere a frase "uma criatura fofa azul percorreu a floresta bela". Tome X como a matriz com as representações em *embeddings* dessa frase. Caso o espaço vetorial dos embeddings possua apenas 3 dimensões, a matriz terá a seguinte forma:

$$X = \begin{bmatrix} \text{uma} & \text{criatura} & \text{fofa} & \text{azul} & \text{percorreu} & \text{a} & \text{floresta} & \text{bela} \\ \hline 0.9 & 0.1 & 0.2 & 0.3 & 0.8 & 0.9 & 0.3 & 0.4 \\ 0.1 & 0.9 & 0.8 & 0.7 & 0.2 & 0.4 & 0.7 & 0.5 \\ 0.2 & 0.2 & 0.7 & 0.6 & 0.0 & 0.1 & 0.3 & 0.1 \end{bmatrix}$$

Em seguida, para calcular a atenção de cada palavra em relação às outras, os seguintes passos são efetuados:

1. Cálculo das matrizes $Query$, Key e $Value$: Durante a etapa de treinamento da rede, matrizes de peso W^Q, W^K e W^V são determinadas de acordo com os dados de treinamento. Essas matrizes servirão para determinar os valores de Q, K e V para uma dada entrada. Ou seja, a matriz de pesos X das palavras da entrada será multiplicada pelos pesos, e isso determinará as matrizes Q, K e V para a entrada X:

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V$$

2. Cálculo das Pontuações de Atenção: Em seguida, calculamos as pontuações de atenção (AS) como o produto escalar de Q e K, normalizado pela raiz quadrada da dimensão dos vetores de K (d_k):

$$AS = \frac{QK^T}{\sqrt{d_k}}$$

De volta ao exemplo, vamos assumir que a Query codifica algo como "adjetivos próximos à posição atual". Com isso, a matriz de atenção irá refletir isso de maneira que adjetivos próximos a uma dada palavra terão um valor maior de atenção para a mesma:

	uma	**criatura**	fofa	azul	percorreu	a	floresta	bela
Uma	0.1	0.2	0.1	0.1	0.2	0.2	0.1	0.1
criatura	0.3	0.1	0.2	0.1	0.2	0.1	0.1	0.2
fofa	0.5	**0.7**	0.8	0.9	0.4	0.1	0.1	0.1
azul	0.3	**0.7**	0.8	0.9	0.6	0.3	0.2	0.1
percorreu	0.1	0.1	0.2	0.2	0.2	0.1	0.1	0.2
a	0.1	0.1	0.3	0.2	0.2	0.1	0.2	0.2
floresta	0.1	0.1	0.3	0.2	0.2	0.1	0.1	0.1
bela	0.2	0.2	0.1	0.2	0.5	0.9	0.8	0.8

Note, por exemplo, como para a palavra "criatura", os valores das linhas 'fofa" e "azul" são elevados. A palavra " bela", apesar de ser um adjetivo, não recebe uma atenção tão grande a partir da palavra "criatura" devido à sua distância.

3. Aplicação das Pontuações de Atenção ao $Value$: As pontuações de atenção são normalizadas por uma operação de $softmax$ para obter as probabilidades de atenção, e então multiplicadas pela matriz V para obter as representações finais ponderadas.

$$\text{Attention Output} = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Por exemplo, se "floresta" tiver uma alta atenção em relação a "bela", a representação final de "floresta" incorporará informações importantes dessas palavras. Esse processo é repetido para cada palavra na sequência e resulta numa representação final que ajusta a direção desse vetor para uma região mais próxima ao conceito ponderado de todas essas palavras.

Finalmente, vale ressaltar que na prática, existem diferentes formas de relacionar termos através de diferentes óticas de comparação. Portanto, em modelos reais é empregado um mecanismo de *multi-headed attention*, que permite ao modelo capturar diferentes tipos de relacionamentos em diferentes subespaços de representação. Em vez de aplicar uma única função de atenção, o *Transformer* utiliza múltiplas "cabeças" de atenção, onde cada cabeça realiza uma operação de atenção separada e independente. Os resultados dessas várias atenções são então concatenados e projetados através de uma camada linear, que os retorna ao espaço dimensional original da entrada como na figura 3.4. Isso permite que o modelo aprenda e integre in-

formações de diferentes perspectivas simultaneamente, aumentando a riqueza das representações contextuais.

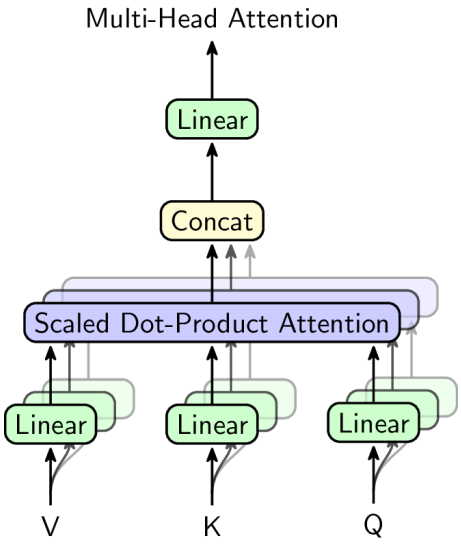


Figura 3.4: Diagrama de arquitetura multi-headed attention

Ao usar *multi-headed attention*, esses cálculos são feitos em paralelo em várias subespaços, e os resultados são concatenados para formar a representação final, proporcionando uma visão mais detalhada e multi-dimensional da entrada.

3.4 Engenharia de Prompt

No contexto de LLMs, um *prompt* é uma entrada de texto fornecida ao modelo para orientar sua geração de saída. O *prompt* define o contexto e pode ser uma pergunta, uma instrução ou uma sequência de palavras que o modelo deve continuar ou responder.

Um exemplo de *prompt* onde é realizada uma pergunta a um modelo LLM generalista:

```
PROMPT:
Descreva em uma linha modelos LLM.

RESPOSTA:
Modelos LLM (Large Language Models) são sistemas de inteligência
artificial projetados para processar e gerar linguagem natural com
base em grandes quantidades de dados textuais pré-existentes.
```

A eficácia do *prompt* é crucial, pois influencia diretamente a relevância e a precisão da resposta do modelo conforme figura 3.5. A arte de formular *prompts* eficazes

é conhecida como engenharia de *prompt*, que envolve criar e ajustar *prompts* para obter as respostas desejadas do modelo.

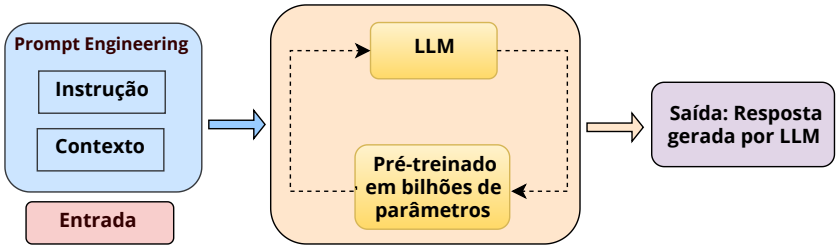


Figura 3.5: Fluxo de aplicação da engenharia de *prompt* como processamento da entrada do usuário

A engenharia de *prompts* emergiu como uma técnica crucial para aprimorar as capacidades dos LLMs. Essa técnica envolve a elaboração estratégica de instruções específicas para tarefas, conhecidas como *prompts*, com o objetivo de orientar a saída do modelo sem a necessidade de alterar seus parâmetros. Foi inicialmente investigada e popularizada no contexto dos LLMs, conforme descrito por (LIU *et al.*, 2023; TONMOY *et al.*, 2024; CHEN *et al.*, 2024).

A importância da engenharia de *prompts* é especialmente evidente na sua capacidade de aumentar a adaptabilidade dos LLMs. Ao permitir ajustes nos resultados do modelo através de instruções cuidadosamente elaboradas, a engenharia de *prompts* possibilita que esses modelos se destaquem em diversas tarefas e domínios, diferentemente dos paradigmas tradicionais que requerem retreinamento ou ajustes finos extensivos para alcançar um desempenho específico.

As técnicas de engenharia de prompt incluem:

- *Prompt Tuning*: Ajustar os prompts com base nas respostas do modelo. Isso pode envolver iterar sobre diferentes formulações até encontrar a mais eficaz (AMATRIAIN, 2024).

- *Zero-Shot Prompting*: Criar prompts claros e específicos para direcionar o modelo de maneira precisa. Evitar ambiguidades e fornecer contexto suficiente é essencial (BROWN *et al.*, 2020).

- *Chain-of-Thought Prompting*: Dividir um problema complexo em etapas menores e fornecer instruções passo a passo no prompt, ajudando o modelo a seguir um raciocínio lógico (WEI *et al.*, 2023).

- *Few-Shot Prompting*: Incluir exemplos no prompt para orientar o modelo sobre o tipo de resposta esperada. Pode ser uma forma eficaz de ensinar o modelo a seguir um padrão específico (BROWN *et al.*, 2020).

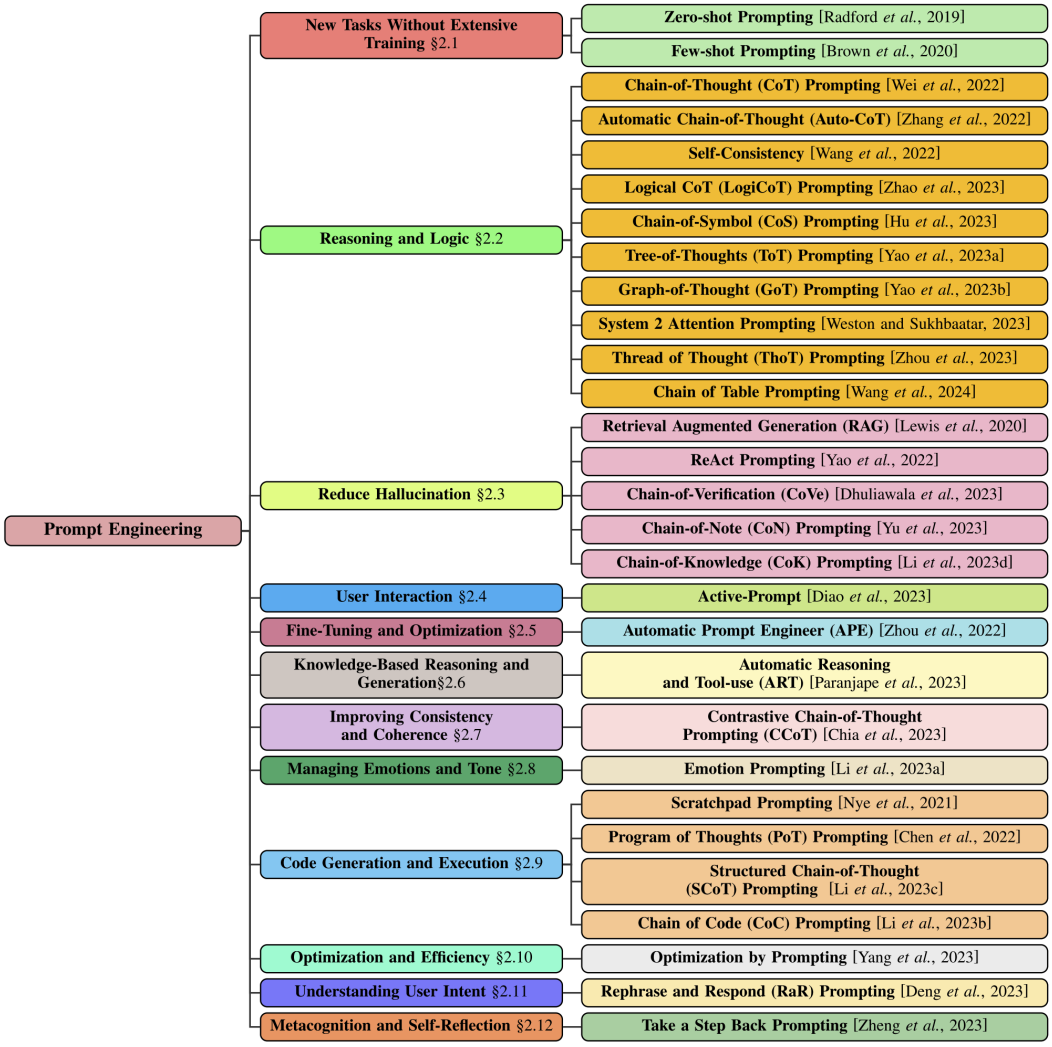


Figura 3.6: Técnicas de *prompt* organizadas por domínios (SAHOO *et al.*, 2024)

- *Prefix-Tuning*: Ajustar prefixos no início dos prompts para otimizar as respostas sem alterar o modelo subjacente (LI e LIANG, 2021).
- *Contextual Prompts*: Fornecer informações adicionais no prompt que possam ajudar o modelo a gerar uma resposta mais informada e relevante (TANG *et al.*, 2022).

Além das principais técnicas mencionadas, em uma pesquisa realizada em fevereiro de 2024, mais de 29 técnicas de *prompts* distintas foram categorizadas, conforme figura 3.6 (SAHOO *et al.*, 2024).

O foco nesse trabalho será nas técnicas de *Chain-of-Thought*, *Zero-Shot Prompting* e *Few-Shot Prompting*. A abordagem de *Chain-of-Thought* foi incluída para investigar se um raciocínio passo a passo pode melhorar a qualidade da classificação, dado o desempenho superior demonstrado por WEI *et al.* (2023); KOJIMA

et al. (2023) em tarefas de uso final comparado às técnicas basais *Zero Shot* e *Few Shot*.

3.4.1 Zero-Shot Prompting

O conceito de *Zero-Shot prompting* refere-se à capacidade dos modelos de linguagem de realizar tarefas para as quais não receberam nenhum exemplo específico de treinamento. Ao contrário do *Few-Shot prompting*, onde alguns exemplos são fornecidos para guiar o modelo, no *Zero-Shot prompting* o modelo deve depender exclusivamente de seu treinamento prévio e de sua capacidade de generalização para entender e responder à nova tarefa. Essa técnica destaca a flexibilidade e a versatilidade dos modelos de linguagem, permitindo que eles abordem uma ampla variedade de tarefas sem a necessidade de dados de treinamento específicos (LIU *et al.*, 2023).

Exemplo de prompt e resposta em *Zero-Shot*:

```
PROMPT:
Traduza a seguinte frase do inglês para o francês:

What time is it?

RESPOSTA:
Quelle heure est-il?
```

3.4.2 Few-Shot Prompting

O conceito de *Few-Shot prompting* foi introduzido para permitir que esses modelos generalizem a partir de um número muito pequeno de exemplos, reduzindo significativamente a necessidade de grandes quantidades de dados de treinamento específicos para cada nova tarefa. O racional por trás dessa técnica é que, ao fornecer apenas alguns exemplos, o modelo pode capturar o padrão ou a estrutura da tarefa e aplicar esse entendimento a novos casos. Não há um número ideal de exemplos bem estabelecido na literatura, com alguns autores mencionando algo entre 4 e 8 (MIN *et al.*, 2022) como o ponto ideal, e outros algo entre 10 e 100 (BROWN *et al.*, 2020).

Exemplo de *prompt* e resposta em *Few-Shot*:

```
PROMPT:
Considere os seguintes exemplos de traduções do inglês para o francês:

Hello, how are you? -> Bonjour, comment ça va?
I love to learn new languages. -> J'adore apprendre de nouvelles
langues.
```

```
What is your name? -> Quel est votre nom?

Traduza a seguinte frase:

What time is it?

RESPOSTA:
What time is it? -> Quelle heure est-il?
```

3.4.3 Chain-of-Thought

CoT (Chain-of-Thought) é uma técnica de *prompt* que visa melhorar a capacidade dos modelos de linguagem de resolver problemas complexos que requerem raciocínio sequencial (SAHOO *et al.*, 2024). A técnica foi introduzida para abordar as limitações dos modelos tradicionais que, apesar de seu poder de geração de texto coerente e contextualmente relevante, muitas vezes falham em manter uma linha de raciocínio consistente ao longo de várias etapas de um problema. Ao estruturar a geração de texto como uma sequência de pensamentos encadeados, CoT facilita o processamento de tarefas que demandam lógica e passo-a-passo.

Exemplo de prompt e resposta em *Chain-of-Thought*:

```
PROMPT:
Classifique a seguinte mensagem de chat do Dota 2 como 'Tóxica' ou
'Não-Tóxica', raciocinando sobre o conteúdo passo a passo:
Mensagem: "Bom dia a todos!"
Passo 1: Determine se a mensagem contém ataques pessoais ou insultos.
Passo 2: Avalie se a linguagem usada é ofensiva ou depreciativa.
Passo 3: Conclua se o tom geral da mensagem é hostil ou negativo.
Retorne apenas a Classificação.

RESPOSTA:
Não-Tóxica.
```

No próximo capítulo, definiremos a metodologia para a escolha e implementação de LLMs em uma tarefa de uso final, considerando um conjunto estruturado de etapas. Definiremos também um modelo matemático para análise de impacto no negócio do LLM selecionado.

Capítulo 4

Método de escolha de LLMs

Neste capítulo, delineamos a metodologia para a escolha de LLMs em uma tarefa de uso final, seguindo um conjunto estruturado de etapas. Inicialmente, definimos o cenário para identificar o contexto e os desafios. Em seguida, determinamos a tarefa da IA *e.g.* classificação, sumarização, tradução, especificando os objetivos e requisitos funcionais. Por fim, será definida a base de testes, selecionando e preparando os conjuntos de dados necessários para o treinamento e avaliação dos modelos.

Na fase de elaboração de *prompts*, desenvolvemos exemplos e instruções que orientarão os modelos na execução da tarefa definida, usando técnicas de Engenharia de *Prompt*. A implementação é planejada, estabelecendo a infraestrutura e recursos computacionais necessários. Em seguida, diferentes LLMs são selecionados e avaliados quanto à sua adequação ao problema em questão, sendo então executados e aplicados às tarefas definidas, com os resultados coletados para análise. Os modelos são avaliados utilizando métricas de desempenho relevantes a tarefa, como *precision*, *recall* e *F1-score*.

Adicionalmente, propomos nesse trabalho um modelo para análise de impacto do uso do LLM selecionado no negócio, considerando os custos e benefícios associados ao uso dos LLMs. Finalmente, escolhe-se o LLM mais adequado, garantindo uma solução eficaz e economicamente viável para a tarefa.

4.1 Metodologia

Na figura 4.1, podemos ver um diagrama *Business Process Model and Notation* (BPMN) (VON ROSING *et al.*, 2015) da metodologia proposta. Inicia-se o processo definindo o cenário do experimento, determinando a tarefa e o objetivo principal da pesquisa, seguido do levantamento de recursos e definição de base de dados apropriada. Após, define-se a implementação, e são selecionados modelos candidatos.

A próxima etapa é a elaboração de *prompts* usando estratégias relevantes de Engenharia de *Prompt*, seguida da execução e avaliação dos modelos, avaliando

desempenho e custo. Em seguida, é feita uma análise de impacto estimado no negócio, finalizando na escolha do LLM apropriado para a pesquisa.

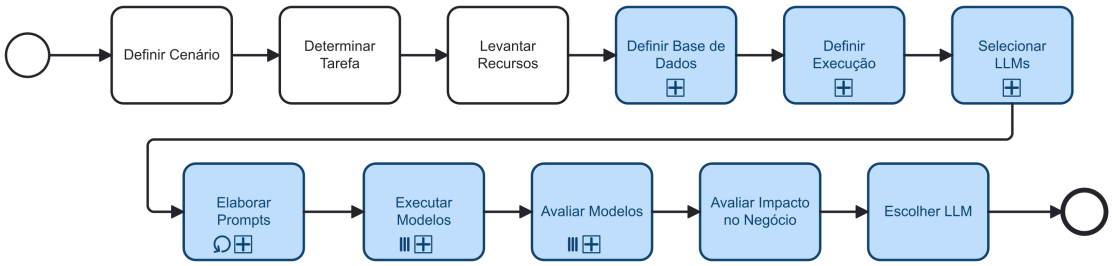


Figura 4.1: Diagrama BPMN do processo. Em *Definir Base de Dados*, o símbolo de + significa um subprocesso. Em *Elaborar Prompts*, o ícone representa um *loop*, e em *Executar Modelos* e *Avaliar Modelos*, os processos podem ser executados em paralelo para cada modelo.

Na próxima seção discutiremos as etapa do experimento, com suas entradas e saídas esperadas. Dada a natureza e escopo desse trabalho, daremos um maior enfoque nas etapas destacadas no diagrama. Para melhor visualização, o diagrama completo expandido encontra-se disponível no anexo A.

4.1.1 Definir Cenário

Nesta etapa busca-se definir o cenário. O cenário fornece o contexto necessário para compreensão do problema a ser resolvido e como as soluções propostas aplicam-se ao contexto. Conceitos de negócios, como o *Increase Revenue, Avoid Costs e Improve Service* (IRACIS), podem ser usados para auxiliar na tomada de decisão, justificando e orientando investimentos, sendo usados para avaliar e comunicar valor de um projeto ou investimento (RUBLE, 1997; GANE e SARSON, 1979).

Descrever um cenário envolve criar uma imagem clara e detalhada do contexto em que um problema ou estudo está inserido. Isso inclui identificar os principais elementos e suas interações, bem como a partes interessadas, objetivos de negócio e métricas de sucesso (GANE e SARSON, 1979):

- **Contexto:** Descrição do ambiente onde o sistema ou projeto será utilizado.
- **Partes Interessadas:** Abrange pessoas, grupos ou organizações interessadas ou afetas pelo cenário. A identificação das partes interessadas leva a compreender diferentes perspectivas e necessidades que devem ser consideradas.
- **Objetivos de Negócio:** Define os objetivos que a organização ou projeto espera alcançar.

- **Métricas de Sucesso:** Define como o sucesso das soluções será medido. Pode ser medido através de métricas quantitativas, com dados de performance e *Key Performance Indicator* (KPI) ou qualitativas, como *feedback* de usuários.

Ao aplicar o IRACIS, cada problema e oportunidade é, então, convertido em uma frase objetiva. Isso permite que os objetivos sejam avaliadas com base em seu impacto no aumento de receita, redução de custos e melhoria do serviço ao cliente, garantindo que os esforços técnicos estejam focados em benefícios de negócios tangíveis.

Tomando como exemplo uma empresa de *e-commerce* planejando implementar um novo sistema de recomendação de produtos utilizando o IRACIS para auxiliar na tomada de decisão, o problema é definido como: A empresa enfrenta desafios relacionados a baixa eficácia das recomendações aos clientes, principalmente as personalizadas, levando a baixa satisfação e fidelização dos clientes. Após aplicação do IRACIS, podemos definir um objetivo claro, conforme tabela 4.1.

Tabela 4.1: Componentes de análise com *IRACIS*

Componente	**Aplicação**
Aumentar Receita	Esperar um aumento nas vendas devido a recomendações mais precisas e personalizadas que incentivam compras adicionais.
Evitar Custos	Reduzir custos operacionais usando um sistema mais robusto e com desempenho superior
Melhorar Serviço	Proporcionar uma experiência de compra mais agradável e personalizada para os clientes, aumentando a satisfação e a lealdade.
Objetivo	Aumentar a receita ao implementar um sistema de recomendação mais preciso e personalizado, que incentive compras adicionais e proporcionar uma experiência de compra mais agradável e personalizada ao cliente.

4.1.2 Determinar Tarefa

Nesta etapa, busca-se especificar claramente a tarefa a ser executada e os subobjetivos da pesquisa. A definição deve ser feita dado a compreensão do cenário e as expectativas das partes interessadas. Deve-se identificar os subproblemas e tarefas específicas que podem ser resolvidos com cada problema sendo claramente definido em termos de saída e tipo de tarefa, conforme visto em tabela 4.2. Com isso, pode-se definir uma solução composta de uma ou múltiplas subtarefas, com o intuito de resolver o problema da pesquisa.

Tabela 4.2: Classes das tarefa de classificação

Tarefa	**Saída**
Classificação de Toxicidade	Tóxico, Não Tóxico
Análise de Sentimentos	Positivo, Negativo, Neutro
Tradução de Textos	Texto Traduzido

Estabelecido os requisitos do negócio, a solução deve ser técnica e economicamente viável. Isso implica na construção de um arcabouço experimental capaz de metrificar e analisar a solução dado as métricas de sucesso estabelecidas, *e.g. Return on Investment* (ROI), maior desempenho na tarefa proposta, busca do estado da arte, qualidade ou velocidade de resposta.

4.1.3 Levantar Recursos

Nesta etapa, busca-se identificar e assegurar todos os recursos necessários para a pesquisa, visando garantir que se disponha de todos os elementos necessários para sua execução. Primeiramente, avaliam-se os recursos computacionais, considerando a necessidade de servidores, *Graphics Processing Unit* (GPUs)) especializadas e infraestrutura de armazenamento adequada para suportar tanto o treinamento quanto a inferência dos modelos. A escolha de provedores de nuvem, como *AWS*[1], *Google Cloud*[2] ou *Azure*[3], pode ser considerada para garantir escalabilidade e flexibilidade.

4.1.4 Definir Base de Dados

Nesta etapa, busca-se definir uma base de teste para uso na solução. O processo sistemático e iterativo, ilustrado pelo diagrama na figura 4.2, visa assegurar a escolha de uma base de dados que atenda aos requisitos específicos do experimento.

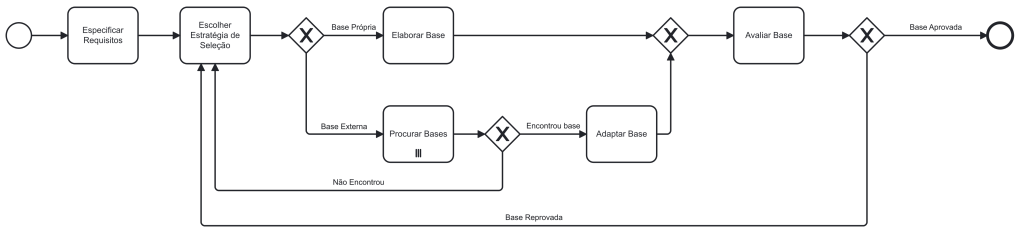


Figura 4.2: Processo de definição de base de dados

Primeiramente, especificam-se os requisitos, definindo os critérios e necessidades do experimento, incluindo as características dos dados necessários, como formato,

[1] `https://aws.amazon.com/`
[2] `https://cloud.google.com/`
[3] `https://azure.microsoft.com/`

volume, qualidade e relevância para a tarefa específica que o LLM deve desempenhar. Com os requisitos definidos, escolhe-se a estratégia de seleção. Nesta fase, decide-se se a base de dados será criada internamente ou procurada em fontes externas, considerando a disponibilidade de dados e os recursos necessários para coletar, rotular ou adquirir essas bases.

Num cenário onde se opta pela criação de uma base própria, elabora-se esta, coletando, organizando e rotulando os dados conforme os requisitos especificados. Dado um conjunto de dados não rotulados, o uso de *crowdsourcing* para rotulamento é considerado uma alternativa viável (WELD *et al.*, 2021; ZHANG *et al.*, 2018). A criação da base de dados pode utilizar de técnicas como *data-augmentation* visando melhorar o desempenho dos modelos através da diversificação dos exemplos de treinamento sem a necessidade de coletar mais dados, considerado o custo e esforço envolvidos (DING *et al.*, 2024).

Alternativamente, caso se decida pela busca de bases externas, recursos como *Kaggle*[4], *UC Irvine Machine Learning* Repository[5] e *Google Dataset Search Engine*[6] podem ser utilizados na busca por bases existentes que atendam aos critérios definidos. Ao encontrar uma base externa adequada, adapta-se à base, caso necessário, o que pode incluir transformações de formato, normalização dos dados e adição de anotações específicas, conforme pode ser visto na tabela 4.3.

Tabela 4.3: Exemplos de Entradas e Rótulos de algumas tarefas de uso final

Tarefa	**Entrada**	**Rótulo**
Classificação de Toxicidade	fk u you ruined my 16 k/d/a	Tóxico
Análise de Sentimentos	*AGI* está cada vez mais próximo	Tristeza

4.1.5 Elaborar Prompts

Nesta etapa busca-se elaborar os *prompts* para a tarefa definida, conforme figura 4.3. Utiliza-se a Engenharia de *Prompt* para elaborar *prompts* que maximizem o potencial dos modelos (SHI *et al.*, 2024).

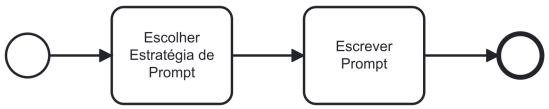


Figura 4.3: Processo de seleção de prompts

Para a elaboração dos *prompts*, deve ser feita uma análise das técnicas e estruturas, como *Zero-Shot*, *Few-Shot* (BROWN *et al.*, 2020), *Chain of Thought* (*CoT*)

[4] `https://www.kaggle.com/datasets`
[5] `https://archive.ics.uci.edu/`
[6] `https://datasetsearch.research.google.com/`

(WEI *et al.*, 2023), *Prompt Space* (SHI *et al.*, 2024), *Automatic Prompt Engineer* (*APE*) (ZHOU *et al.*, 2023), dentre outros. Por ser uma disciplina emergente e em constante evolução, recursos como o *PromptingGuide*[7] e *OpenAI Prompt Engineering Guide*[8] podem ser valiosos na seleção dessas estratégias.

Dada a seleção do conjunto de técnicas a ser testado, formula-se e testa-se os diferentes tipos de *prompts* de forma a atender às necessidades da tarefa em questão, ajustando-os com base no *feedback* e nos resultados das avaliações, buscando refinar a eficácia do modelo. A Engenharia de *Prompt* é uma alternativa flexível e simples de se obter um ganho de desempenho ou iterar em novas estratégias no futuro.

Modelos de código aberto são comumente lançados em pares, com uma versão base e uma *instruct e.g. Llama-3-8b e Llama-3-8b-Instruct, Gemma-2b e Gemma-2b-Instruct*. Recomenda-se o uso das variações *instruct* no processo de experimentação, pois são treinadas em seguir instruções e demonstram desempenho superior aos modelos base em tarefas específicas (OUYANG *et al.*, 2022).

4.1.6 Definir Execução

Nesta etapa, busca-se definir a execução da tarefa definida. Esta fase abrange duas subetapas: definir o algoritmo e definir as métricas de avaliação, conforme diagrama na figura 4.4. Aqui, elabora-se os programas e *scripts* necessários para execução da tarefa, considerando a coleta de dados adequada para análise das métricas definidas.

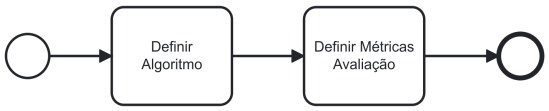


Figura 4.4: Processo de Definir Execução de LLMs

Na subetapa de definir o algoritmo, estabelece-se a base tecnológica necessária para a execução dos modelos. Isso inclui a seleção de hardware, software e recursos computacionais essenciais. Primeiramente, especifica-se a infraestrutura, com o uso de servidores de alto desempenho ou provedores de nuvem. Além disso, definem-se os ambientes de desenvolvimento e execução, incluindo *frameworks* e bibliotecas, *e.g. TensorFlow*[9], *PyTorch*[10] e *Hugging Face Transformers*[11], *SageMaker Python SDK*[12]. A definição de parâmetros, como a temperatura, deve ser definida nesta etapa de forma a atender a tarefa. Em tarefas de classificação, buscam-se resultados

[7] `https://www.promptingguide.ai/`
[8] `https://platform.openai.com/docs/guides/prompt-engineering`
[9] `https://www.tensorflow.org/`
[10] `https://pytorch.org/`
[11] `https://github.com/huggingface/transformers`
[12] `https://sagemaker.readthedocs.io/`

mais determinísticos nas saídas do modelo, o que leva a uma temperatura próxima de zero.

Na subetapa de definir as métricas de avaliação, estabelecem-se critérios claros para medir o desempenho dos LLMs. As métricas de avaliação são essenciais para garantir que os modelos atendam aos objetivos definidos. No caso do uso dos LLMs como classificador, por exemplo, métricas como precisão, *recall*, *F1-score* são frequentemente utilizadas para avaliar a eficácia do modelo. Essas métricas permitem uma análise quantitativa do desempenho, identificando pontos fortes e áreas que necessitam de melhorias.

Além disso, considera-se a inclusão de métricas específicas para avaliar o impacto do modelo no contexto do negócio, como tempo de resposta e custo operacional. Nesta etapa, é importante estabelecer também como será feita a coleta de dados e as saídas, visando o uso dos mesmos na etapa 4.1.9.

4.1.7 Selecionar LLMs

Nessa etapa, busca-se selecionar modelos candidatos a avaliação de desempenho através do uso de um critério de seleção. Este processo divide-se em duas fases principais: uma fase inicial de avaliação simplificada e uma fase subsequente de priorização detalhada, conforme pode ser visto na figura 4.5. Abaixo, descrevemos o processo metodológico para a seleção e priorização de LLMs candidatas para a tarefa definida.

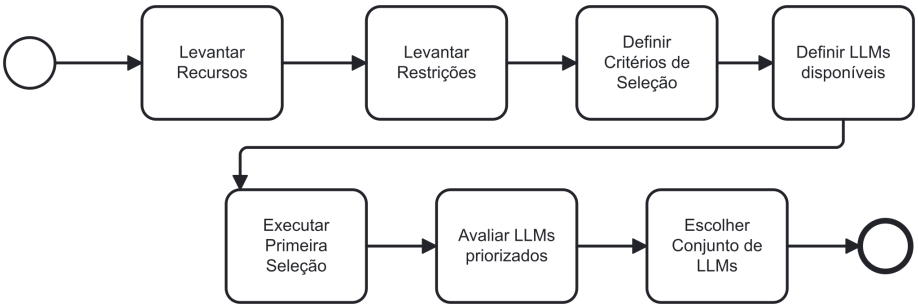


Figura 4.5: Processo de Seleção de LLMs candidatos

A primeira etapa da seleção de LLMs candidatos envolve a identificação e catalogação dos recursos disponíveis para o projeto. Esses recursos podem incluir hardware, *e.g.* GPUs, *Tensor Processing Unit* (TPUs), instâncias em nuvem, software, *e.g.* bibliotecas, *frameworks*, modelos, dados e equipe técnica. Em seguida, identificam-se as restrições que possam impactar a seleção das LLMs. Elas podem ser de natureza variada, incluindo:

- *Custo*: Orçamento disponível para treinamento, implementação e manutenção dos modelos;

- *Tempo*: Prazo disponível para a implementação do projeto;
- *Disponibilidade*: Acessibilidade a determinados modelos ou tecnologias, considerando licenças e compatibilidades.

Dado o vasto número de LLMs disponíveis, é importante definir critérios para seleção, que vão desde tamanho a idiomas suportados pelo modelo. Na próxima seção serão discutidos os possíveis critérios.

Possíveis Critérios de Seleção

Os critérios de seleção devem ser claramente definidos com base em uma avaliação qualitativa de artigos científicos e *benchmarks* disponíveis na literatura. Critérios comuns incluem tamanho, arquitetura do modelo, desempenho, custo, licença e fornecedor.

O tamanho do modelo é um critério crucial, pois modelos maiores tendem a oferecer melhor desempenho, mas exigem mais recursos computacionais e aumentam os custos de operação (BROWN *et al.*, 2020). Portanto, é necessário equilibrar o desempenho esperado com os recursos disponíveis, evitando escolhas que possam comprometer a viabilidade do projeto devido a limitações de hardware ou orçamento.

Além do tamanho, o custo é outro critério fundamental. Isso inclui não apenas o custo de treinamento e inferência, mas também o custo de manutenção do modelo. A arquitetura do modelo também deve ser considerada, uma vez que diferentes arquiteturas têm diferentes capacidades e podem ser mais ou menos adequadas para tarefas específicas. No caso da arquitetura *Transformers*, existem variações e avanços arquiteturais em modelos mais recentes, com diferenças desde a etapa de treinamento até a inferência (HUANG *et al.*, 2024).

Por fim, o fornecedor e a licença do modelo também são fatores preponderantes na escolha do modelo. Modelos disponibilizados através de APIs como *GPT-4-Omni* e *Claude Opus*, podem possuir melhor suporte técnico e ferramentas integradas a um custo mais alto. Modelos de código aberto podem oferecer maior flexibilidade e menor custo, mas podem exigir disponibilidade em serviços de *cloud computing* e mais suporte interno, considerando-se a viabilidade de execução em produção e não apenas para experimentação. A licença do modelo deve ser adequada às necessidades do projeto, garantindo conformidade legal e operacional.

Com os critérios de seleção definidos, listam-se as LLMs disponíveis que poderiam potencialmente atender aos requisitos do projeto. Esta lista pode incluir modelos disponibilizados por *APIs*, e modelos *open-source*, como *Llama-3*,*Gemma* e *Mistral*.

Primeira Seleção

Uma primeira seleção deve ser realizada aplicando critérios e restrições mais simples para filtrar rapidamente as LLMs que não atendem aos requisitos básicos do projeto, *e.g* número de parâmetros, idiomas suportadas. Esta fase visa reduzir o número de modelos candidatos a um conjunto mais gerenciável. Esses modelos devem ser priorizados com base em sua adequação inicial aos critérios e restrições estabelecidos. A priorização pode ser feita através de uma matriz de decisão ou abordagem similar.

Uma avaliação detalhada de cada LLM prioritária deve ser conduzida, considerando todos os critérios de seleção definidos. Esta avaliação pode envolver experimentos empíricos, testes em conjuntos de dados específicos e análises de *benchmarks*, quando aplicável.

Escolher Conjunto de Candidatos

Com base nas avaliações detalhadas, o conjunto final de LLMs que melhor atendem aos critérios e restrições do projeto deve ser selecionado. Este conjunto de modelos será então utilizado nas etapas subsequentes. Este processo sistemático assegura que a seleção de modelos candidatos seja fundamentada em análises e alinhada com os objetivos e restrições do projeto.

4.1.8 Executar Modelos

Esta etapa visa a execução dos modelos selecionados definidos na etapa anterior. Conforme diagrama na figura 4.6, avaliam-se as adaptações para execução do modelo, necessário para adequar a execução aos requisitos individuais. Os requisitos podem variar em parâmetros, infraestrutura, em fornecedor em tipo de serviço, *e.g.* no caso de modelos disponíveis através de APIs, dentre outros.

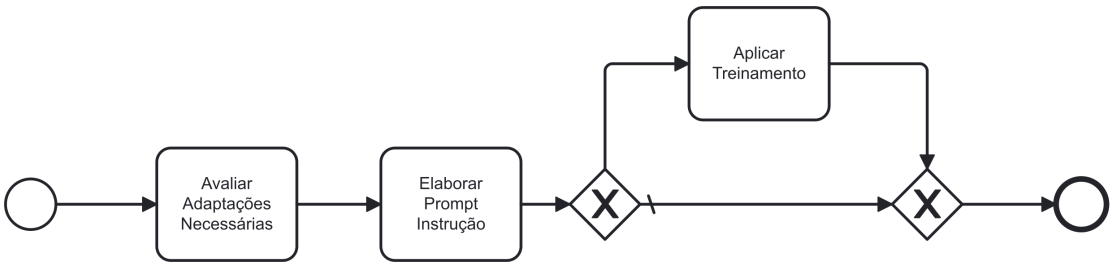


Figura 4.6: Processo de Avaliação de modelos

Em seguida, ajustam-se os *prompts* para o formato de instruções específico de cada modelo. Nesta fase, os *prompts* previamente elaborados são adaptados para atender aos requisitos de entrada do modelo escolhido. Cada modelo pode ter um

formato particular para receber instruções, e a adaptação dos *prompts* garante que o modelo compreenda e processe corretamente as entradas fornecidas.

Após o ajuste dos *prompts*, passa-se para a fase de decisão sobre a aplicação de *fine-tuning*. Nesta etapa, avalia-se se o *fine-tuning* é necessário com base nos resultados preliminares do modelo. O *fine-tuning* envolve treinar o modelo com dados específicos do domínio para melhorar sua precisão e relevância para a tarefa.

Aplicar o *fine-tuning* é geralmente mais custoso em comparação com o uso do modelo *out-of-the-box*, exigindo recursos computacionais substancialmente maiores e custo de manutenção para retreinamento contínuo do modelo (SHI *et al.*, 2024). A escolha entre *fine-tuning* e uso direto do modelo depende do equilíbrio entre os recursos disponíveis e a necessidade de obter uma maior precisão para a tarefa específica.

4.1.9 Avaliar Modelos

Nessa etapa, executa-se uma análise detalhada de custo e desempenho dos modelos selecionados, facilitando a tomada de decisão sobre qual modelo será implementando na solução, conforme figura 4.7.

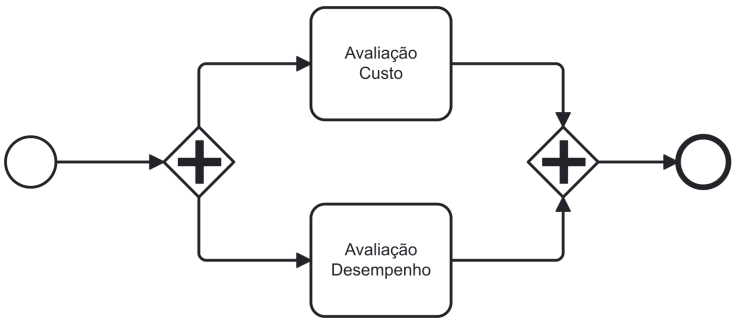


Figura 4.7: Processo de Execução de modelos

Na avaliação de custo, estima-se todos os custos associados à utilização e execução dos modelos. Isso inclui custos computacionais, como consumo de energia, tempo de processamento e utilização de servidores e GPUs, além de despesas com armazenamento de dados ou com plataformas de computação em nuvem, ou *tokens* consumidos, no caso de APIs como *OpenAI* ou *Amazon Bedrock*.

Paralelamente, realiza-se a avaliação de desempenho dos modelos. Esta avaliação envolve a análise das métricas definidas na etapa 4.1.6, para medir o desempenho dos LLMs em realizar a tarefa específica.

4.1.10 Análise de Impacto para o Negócio

Esta etapa envolve a análise de impacto para o negócio do LLM selecionado e como essa escolha influenciará os aspectos econômicos e operacionais da organização, assegurando que o modelo traga benefícios tangíveis e alinhados aos objetivos estratégicos. Para tal, propomos um modelo, visando orientar a tomada de decisão e, principalmente, mensurar o impacto do erro do modelo na receita final do negócio, e em última instância ajudando a responder quando se deve usar um modelo mais custoso, mas com desempenho superior sobre um modelo menos custoso.

O modelo é baseado principalmente na satisfação, como principal dos pilar da retenção, sendo proporcional a retenção do usuário. Inicialmente, modelamos o impacto das classificações, baseado nas probabilidades de saídas da tarefa de classificação. Conforme figura 4.8, modelamos de forma que:

- *Verdadeiros Positivos* (TP): impactam em reduzir a quantidade de conversas tóxicas, afetando a toxicidade e reduzindo o descontentamento dos usuários do jogo, impactando positivamente na retenção.
- *Falsos Negativos* (FN): conversas tóxicas deixariam de estar sendo bloqueadas, o que afetaria o número de conversas de maneira negativa, mantendo seu número mais elevado, aumentando o descontentamento dos jogadores e impactando negativamente na retenção.
- *Falsos Positivos* (FP): conversas não tóxicas bloqueadas afetariam diretamente no descontentamento do usuário, sendo impactado por uma punição incorretamente, impactando na retenção do usuário
- *Verdadeiros Negativos* (TN): não afetam de maneira direta o contentamento, ou quantidade de conversas tóxicas ou não bloqueadas.

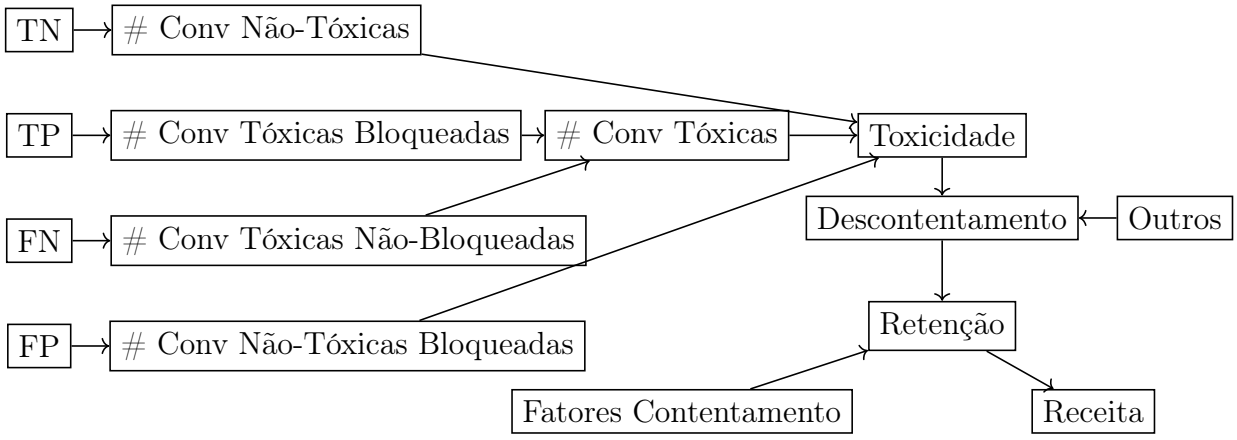


Figura 4.8: Impacto dos resultados da classificação na retenção

Na figura 4.9, podemos ver de forma simplificada o impacto dos resultados da classificação na toxicidade, segundo o modelo proposto. Discutiremos de forma mais profunda o impacto das probabilidades dos resultados de classificação na toxicidade na seção 4.1.10.

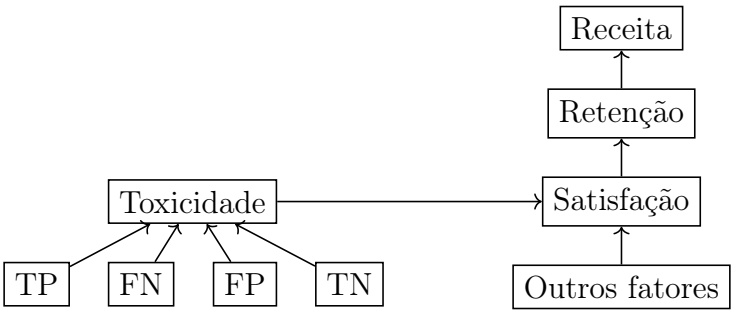


Figura 4.9: Versão simplificada do impacto dos tipos de classificação na retenção

Além da retenção, a redução da toxicidade e a qualidade da classificação deve ser ponderada em relação aos custos de processamento das conversas. Ressaltamos que o modelo apresentado usa como regra geral de simplificação considerar que todas as funções são lineares.

Receita em função da retenção

Sendo I a receita (*income*) da empresa de jogos, assumimos que a receita é diretamente proporcional a retenção do usuário, R. Logo, podemos expressar a receita como uma função da retenção:

$$I = f(R), \tag{4.1}$$

onde, por simplicidade, assumiremos que f é uma função linear. Logo:

$$f(R) = k_r R, \tag{4.2}$$

onde k_r é uma constante de representando a receita gerada por usuário retido. Essa receita pode ser determinada a partir de valores como LTV (Life Time Value), que são normalmente conhecidos pela organização.

Retenção em função da Satisfação do Usuário

A retenção de usuários R é influenciada pela satisfação do usuário S. Expressamos a retenção como uma função da satisfação do usuário:

$$R = g(S), \tag{4.3}$$

onde, por simplicidade, assumiremos que g é uma função linear. Logo, podemos definir:

$$g(S) = k_s S, \tag{4.4}$$

onde k_s é a constante representando a taxa de retenção por unidade de satisfação do usuário. Esse número deve ser definido pelas empresas, ou pelo menos seu complemento, como o *churn*, que mede a taxa de clientes que cancelaram ou pararam de consumir os produtos ou serviços de uma empresa em um determinado período de tempo.

Satisfação do usuário em função da toxicidade e classificação incorreta

A satisfação do usuário S é afetada pelo nível de toxicidade T nas conversas e pela taxa de classificação incorreta M, ou seja, conversas válidas tratadas como tóxicas. O M inicial é zero, pois essa taxa não existe inicialmente, já que supomos que não é feito o tratamento que elimina as conversas tóxicas.

Expressamos a satisfação do usuário como:

$$S = h(T, M), \tag{4.5}$$

por simplicidade, assumimos que h é uma função linear. Logo, escolhemos:

$$h(T, M) = k_s - k_t T - k_m M, \tag{4.6}$$

onde k_s representa a satisfação geral com o produto, k_t é o impacto negativo por unidade de toxicidade e k_m é o impacto negativo por unidade de classificação incorreta. Esses valores não são conhecidos normalmente pela empresa, devendo ser estimados caso a caso, ou estudados na forma de cenários.

Variação na receita devido à aplicação do LLM

A variação da renda ΔI devido à aplicação do LLM pode ser modelada considerando o impacto na satisfação e retenção do usuário, menos o custo da operação:

$$\Delta I = k_r \Delta R - \bar{C}_T, \tag{4.7}$$

onde ΔR é a variação na retenção devido a mudanças na satisfação do usuário ΔS:

$$\Delta R = k_s \Delta S. \tag{4.8}$$

A mudança na satisfação do usuário ΔS é afetada pelas mudanças nas taxas de toxicidade ΔT e classificação incorreta M:

$$\Delta S = -k_t \Delta T - k_m M. \tag{4.9}$$

Assumindo que o LLM reduz a toxicidade em uma constante α e cria uma taxa de classificação incorreta M, temos:

$$\Delta T = -\alpha T. \tag{4.10}$$

Combinando essas equações, obtemos:

$$\Delta S = -k_t(-\alpha T) - k_m M, \tag{4.11}$$

$$\Delta S = k_t \alpha T - k_m M. \tag{4.12}$$

Portanto, a variação na retenção é:

$$\Delta R = k_s(k_t \alpha T - k_m M). \tag{4.13}$$

E a variação em receita é:

$$\Delta I = k_r k_s(k_t \alpha T - k_m M) - \bar{C}_T. \tag{4.14}$$

Cálculo de Probabilidades

A eficácia do LLM na classificação de conversas pode ser avaliada através das probabilidades de verdadeiros positivos (TP), verdadeiros negativos (TN), falsos positivos (FP), falsos negativos (FN), alucinações positivas (HP) e alucinações negativas (HN). Estas probabilidades têm impacto tanto na satisfação do usuário, quanto no custo de implementação do LLM. As probabilidades na tarefa de classificação com uso de LLMs podem ser definidas como:

- **Falsos Positivos (FP)**: Conversas não tóxicas classificadas incorretamente como tóxicas.
- **Verdadeiros Positivos (TP)**: Conversas tóxicas corretamente classificadas como tóxicas.
- **Falsos Negativos (FN)**: Conversas tóxicas incorretamente classificadas como não tóxicas.
- **Verdadeiros Negativos (TN)**: Conversas não tóxicas corretamente classificadas como não tóxicas.

- **Alucinações Positivas (HP)**: Conversas tóxicas em que o LLM gerou alucinação.
- **Alucinações Negativas (HN**): Conversas não tóxicas em que o LLM gerou alucinação.

Denotamos as probabilidades desses resultados como P_{FP} , P_{TP} , P_{FN}, P_{TN}, P_{HP} e P_{HN}, respectivamente.

$$P_{TP} + P_{TN} + P_{FP} + P_{FN} + P_{HP} + P_{HN} = 1. \tag{4.15}$$

Considerando a fração α e M, temos:

$$\alpha = P_{TP} + P_{FN} + P_{HP} \tag{4.16}$$

e

$$M = P_{FP} + P_{TN} + P_{HN}. \tag{4.17}$$

4.1.11 Impacto na satisfação e retenção do usuário

As probabilidades desses resultados afetam diretamente a satisfação do usuário. Por exemplo, uma taxa mais elevada de falsos positivos (FP) pode levar à insatisfação devido a conversas sendo bloqueadas, enquanto uma taxa mais alta de falsos negativos (FN) pode levar à insatisfação devido ao não bloqueio de conversas tóxicas, conforme pode ser visto na figura 4.10.

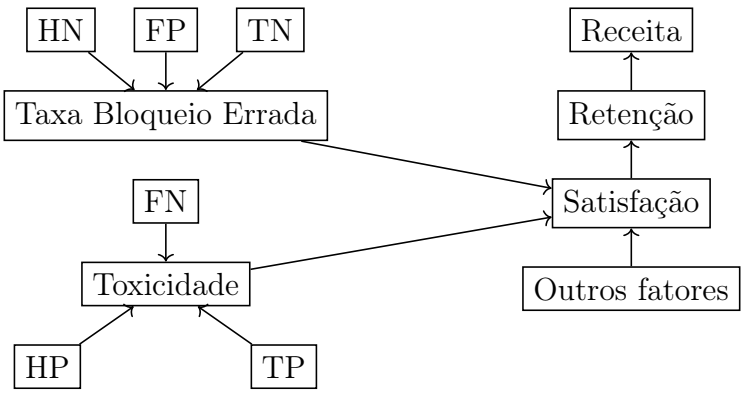


Figura 4.10: Impacto dos tipos de classificação na satisfação do usuário

A variação na satisfação do usuário (ΔS) pode ser expressa como:

$$\Delta S = -k_t T\alpha - k_m M, \tag{4.18}$$

onde T é o nível de toxicidade inicial e M é a taxa inicial de erros de classificação.

4.1.12 Impacto na Receita

A variação global da receita (ΔI) devido à aplicação do LLM é afetada tanto pela mudança na satisfação do usuário, quanto pelo custo de processamento:

$$\Delta I = k_r k_s (k_t \alpha T - k_m M) - \bar{C}_T. \tag{4.19}$$

Substituindo α e M na equação para ΔI, temos:

$$\Delta I = k_r k_s \left[k_t T (P_{TP} + P_{FN} + P_{HP}) - k_m (P_{FP} + P_{TN} + P_{HN}) \right] - \bar{C}_T, \tag{4.20}$$

onde T é a toxicidade inicial, k_r é uma constante representando a receita gerada por usuário retido, k_s é a constante representando a taxa de retenção por unidade de satisfação do usuário, k_t é o impacto negativo por unidade de toxicidade e k_m é o impacto negativo por unidade de classificação incorreta e $\bar{C}_T$ é o custo médio de processamento.

4.1.13 Escolher LLM

Esta etapa finaliza o processo de seleção, assegurando que o modelo escolhido seja o mais adequado para a tarefa em questão. Realiza-se uma análise comparativa entre os modelos candidatos utilizando-se os dados da avaliação de custo e desempenho disponíveis, em conjunto com a análise de impacto para o negócio.

Após a ponderação, procede-se à seleção do modelo mais adequado. Esta seleção é baseada na análise combinada dos resultados de desempenho e custo, visando maximizar o impacto positivo da escolha no negócio. O modelo escolhido deve demonstrar desempenho adequado para a tarefa, mas também ser sustentável e escalável dentro dos limites orçamentários do projeto. A decisão final leva em consideração tanto os resultados quantitativos das avaliações quanto as considerações qualitativas, como a facilidade de integração do modelo na infraestrutura existente e a flexibilidade para futuras adaptações.

Finalmente, a decisão é documentada, incluindo as justificativas para a escolha do modelo específico, as expectativas de desempenho, impacto no negócio e os planos para a implementação, treinamento, quando aplicável, e monitoramento contínuo. Esta abordagem estruturada para a escolha de um LLM visa garantir que a seleção final esteja bem fundamentada, promovendo uma implementação bem-sucedida e alinhada aos objetivos do projeto.

Capítulo 5

Experimento

Este capítulo utiliza a metodologia proposta para classificação de toxicidade em *chats* de jogos online, dado um problema definido por uma hipotética desenvolvedora de jogos com problemas de toxidade em um de seus jogos. Essa metodologia segue o fluxo definido no capítulo 4.

5.1 Definição de Cenário

A definição do cenário inicia com a definição do problema. A definição correta do problema é crucial para que possa ser feita a definição apropriada do cenário, auxiliando na elaboração do escopo de uma solução apropriada para o problema e para o negócio.

A definição hipotética do problema é: "uma desenvolvedora de jogos enfrenta desafios relacionados a diminuição na retenção de novos jogadores. Devido ao crescimento do jogo, o volume de mensagens geradas inviabiliza a análise por moderadores humanos. O aumento de toxicidade nas partidas têm sido uma reclamação constante dos jogadores em mídias sociais e fóruns, levando a um problema de imagem do jogo como um todo". A partir da definição do problema, elaboram-se os componentes do cenário: contexto, objetivos de negócio e métricas de sucesso (GANE e SARSON, 1979). Os componentes do problema definido podem ser observados na tabela 5.1.

Em seguida, foi aplicado o *framework* IRACIS (RUBLE, 1997; GANE e SARSON, 1979), convertendo o problema em frases objetivas, permitindo que as propostas sejam avaliadas com base em seu impacto. Na tabela 5.2, podemos observar as oportunidades dentro do *framework* e um consequente objetivo definido de forma menos abstrata que a definição do problema inicial.

Na próxima seção, definiremos a tarefa a ser executada, dado o ambiente, a base de jogadores e a compreensão do cenário.

Tabela 5.1: Definição de Cenário

Componente	**Descrição**
Contexto	Desenvolvimento de um sistema de detecção de toxicidade para plataformas de jogos online.
Partes Interessadas	Jogadores, desenvolvedores, moderadores, gestores de projeto, proprietários da plataforma de jogos.
Objetivos de Negócio	Reduzir a toxicidade, melhorar a experiência do usuário, aumentar a retenção de jogadores, aumentar a eficiência na moderação, proteger a imagem do jogo/desenvolvedora.
Métricas de Sucesso	Redução na taxa de *churn*, aumento na retenção de jogadores, redução no número de *reports* de toxicidade com o tempo.

5.2 Determinar Tarefa

Para determinar a tarefa e sub tarefas a serem executadas pela solução, analisa-se o ambiente e base de jogadores. Também é crucial listar e definir os tipos de comportamentos tóxicos observados, exemplificando com casos reais para clarificação do problema.

Considerando o cenário proposto por essa pesquisa na seção 1.2, estabelece-se como tarefa o desenvolvimento de um sistema automatizado economicamente viável para classificação de comportamentos tóxicos em jogos online. Isso envolve o uso de LLMs para analisar o conteúdo das interações dos jogadores e identificar comportamentos abusivos, ofensivos ou prejudiciais. Conforme pode ser visto na tabela 5.3, a tarefa principal da pesquisa é um problema de classificação, onde as conversações dos jogadores são avaliadas como tóxicas ou não.

5.3 Mapeamento de Recursos

Para a realização desta pesquisa, foi essencial garantir a disponibilidade de recursos para a execução do mesmo, especialmente em infraestrutura, visto seu escopo e proposta envolvendo LLMs. A maior parte dos recursos necessários foi obtida através de créditos cedidos pela *Amazon Web Services* (AWS), o que possibilitou o acesso a infraestrutura necessária para o processamento de dados e a implementação dos modelos, majoritariamente através da plataforma *AWS SageMaker*.

Apesar de viabilizarem a pesquisa, os créditos na AWS geraram em algumas restrições, implicando no uso de LLMs disponíveis dentro da plataforma *AWS Sagemaker*, que apesar da ampla oferta de modelos, não possuía disponibilidade de

Tabela 5.2: Análise IRACIS do Sistema de Detecção de Toxicidade

IRACIS	**Descrição**
Aumentar Receita	Aumento da retenção de jogadores, atração de novos usuários, venda de recursos premium.
Evitar Custos	Redução do esforço manual, minimização da perda de usuários, prevenção de danos à reputação.
Melhorar Serviço	Proporcionar um ambiente de jogo mais seguro, resposta rápida a comportamentos tóxicos, feedback e ajustes em tempo real.
Objetivo	Aumentar a retenção de jogadores através da implementação de um sistema robusto de classificação de toxicidade, reduzindo o esforço manual de moderação em escala e prevenindo danos à reputação de desenvolvedora, proporcionando um ambiente de jogo mais seguro e repostas mais rápidas a comportamentos tóxicos.

Tabela 5.3: Exemplos de conversações da base de jogadores

Conversação	**Tóxico**
fk u you ruined my 16 k/d/a	✓
but we dive much any other we are losing	✗
can u stfu no one cares remotely about your or your opinion	✓
dam you're full of yoruself produ of playing riki	✓

alguns dos modelos mais recentes[1], como o *PHI-3* [2] e *Gemini-1.5*. O alto custo da infraestrutura para os modelos também teve impacto no escopo da pesquisa, com um custo total acumulado superior a mil dólares. Os créditos foram cedidos pelo Chamada CNPq/AWS Nº 64/2022 – Acesso às Plataformas de Computação em Nuvem da AWS (*Cloud Credits for Research*).

5.4 Bases de Dados

A escolha de uma base de dados pode ser considerado o maior obstáculo em desenvolver classificadores de conteúdo tóxico (HE *et al.*, 2023). O diagrama apresentado na figura 4.2 ilustra o processo de seleção de bases de dados, que inclui a opção de desenvolver uma bases próprias ou utilizar bases externas já existentes.

[1] Em maio de 2024

[2] `https://azure.microsoft.com/en-us/products/phi-3`

Para atender aos requisitos da pesquisa, a base de dados deve conter conversas de jogos online, incluindo gírias e particularidades específicas desse ambiente, além de estar devidamente rotulada para toxicidade. Na tabela 5.4, podemos ver algumas dessa expressões específicas para jogos.

Tabela 5.4: Exemplos de gírias em um jogo de *Dota 2*

Tipo	**Texto**
Personagem	Pudge, Axe, Weaver, Drow Ranger
Específicos do *Dota 2*	Hookshot, Roshan, Aegis
Termos de Jogo	report, gg ez, noob, mid, jungle, creep

No processo de seleção de estratégia, definiu-se que a elaboração de uma base de dados própria seria inviável devido aos recursos limitados e ao tempo necessário para coletar, anotar e validar dados suficientes dentro do contexto do problema, dado que sua elaboração já apresenta um conjunto significativo de desafios. A acessibilidade a *replays*, *chat logs* e informações associadas têm se tornado mais limitada devido a preocupações com privacidade pelos desenvolvedores dos jogos (MARTENS *et al.*, 2016), o que torna a criação de base próprias ainda mais complexa. Com isto, optamos pela busca de bases de dados externas.

Apesar do amplo número de bases de dados disponíveis de toxicidade, as bases encontradas focadas em toxicidade não são específicas para jogos (ZHANG *et al.*, 2018). Para jogos, as opções válidas envolvem *chat logs* do Tribunal de League of Legends, Kaggle[3] ou extraídos via Open Dota API(TEAM, 2024).

Considerando a proposta desta pesquisa, uma das alternativas viáveis seria a utilização de *chat logs* do *Tribunal* como feito por BLACKBURN e KWAK (2014). Entretanto, essa base de dados poderia carregar viés nas análises, visto que todos os *chat logs* no *Tribunal* são candidatos a jogos com algum nível de toxicidade. Com isto, encontramos apenas dois conjuntos de dados públicos: o *OpenDota API Dataset*[4] e *CONDA* (WELD *et al.*, 2021). A base de dados utilizada por BLACKBURN e KWAK (2014) não foi encontrada publicamente.

Tabela 5.5: Conjuntos de dados de jogos online candidatos

Dataset	**Rótulo**	**Sem Viés**
CONDA	✓	✓
OpenDota API Dataset	✗	✓

Na tabela 5.5 pode ser visto que o único *conjunto de dados* que atendeu os requisitos foi o *CONDA Dataset*, visto que o *OpenDota API Dataset* não disponibiliza

[3] `https://www.kaggle.com/`

[4] `https://www.kaggle.com/datasets/devinanzelmo/dota-2-matches`

rótulos, apenas os *chat logs* em si.

5.4.1 CONDA

O conjuntos de dados utilizado nos experimentos foi o CONDA (*CONtextual Dual-Annotated Dataset*) (WELD *et al.*, 2021). O corpus se apresenta na língua inglesa e consiste de 45 mil registros de chat de 1,9 mil partidas de *Dota 2* e foi originalmente criado para classificação conjunta de intenções e análise de preenchimento de *slots*, que classifica individualmente as palavras das sentenças, sendo este duplamente anotado com quatro classes de intenção e seis classes de *slot*. As classes de intenção podem ser vistas na tabela 5.6.

Tabela 5.6: Classes de Intenções no CONDA

Intenção	**Descrição**
E (**Explicit**)	Contém palavras tóxicas explícitas com a intenção de insultar ou humilhar outros, sem necessidade de considerar o contexto, *e.g. 'fuck off'*.
I (**Implicit**)	Toxicidade oculta que não pode ser vista diretamente no texto, mas inferida pelo contexto da conversa, sem palavras tóxicas explícitas, *e.g. 'u are poor dude'*).
A (**Action**)	Contém uma ação como reportar, elogiar, pausar, parar ou sair do jogo.
O (**Other**)	Não se enquadra em I, E ou A. Pode ou não conter palavras tóxicas, incluindo maldições, autodepreciação ou expressões emocionais não direcionadas a outro, *e.g.* '*kill the fucking helicopter*'.

O *corpus* utiliza do conceito de conversação, identificando o início de uma conversação como a primeira fala na partida ou uma fala que ocorre mais de sessenta segundos após a fala anterior, unindo sentenças consecutivas de um único jogador em uma conversação. Segundo WELD *et al.* (2021), conversações em jogos são pontuais e dinâmicas, devido a própria natureza de jogos multijogador *on-line*, levando conversas muito afastadas a perderem seu contexto ou relevância. Na tabela 5.7, podemos ver algumas sentenças da base de dados. No caso de múltiplas sentenças de um mesmo usuário, pode ser visto o marcador $[SEPA]$.

Tabela 5.7: *Exemplos de sentenças com rótulo de Intenções no CONDA*

Sentença	**Intenção**
i killed it	O(ther)
why u wan [SEPA] remake	A(ction)
idiot	E(xplicit)

Dado o propósito original da base de dados, foram feitos dois ajustes: a conversão de rótulos de intenção para as classes de Tóxico ou Não Tóxico, conforme tabela 5.8 e a união de todas as sentenças de uma conversação em uma linha. Partimos da premissa que que se uma das sentenças em uma conversação é tóxica, a conversação pode ser considerada tóxica, pois há presença de toxicidade na mesma.

Tabela 5.8: *Conversão de rótulos de intenção para categorias de toxicidade*

Intenção	**Convertido**
E (Explícito), I (Implícito)	Tóxico
A (Ação), O (Outro)	Não tóxico

Análise da Base de dados

Os autores desse conjunto de dados hospedam uma competição[5] permanente para detecção de linguagem tóxica e reservam 20% da base como base de teste privada. Com isto, utilizamos os 80% públicos da base em nossos experimentos, totalizando 34793 sentenças. Após a agregação das sentenças das conversações, a base de dados passou a ter um total de 10659 conversações. A base de dados é desbalanceada, com uma proporção de $63,2\%$ de conversações tóxicas para $36,8\%$ de conversações não-tóxicas, conforme figura 5.1.

Dado o custo computacional dos experimentos e a possibilidade de *fine-tuning*, a base final foi separada num *split* 70-30, onde utilizamos 30% da base de dados nos experimentos.

[5] `https://codalab.lisn.upsaclay.fr/competitions/7827`

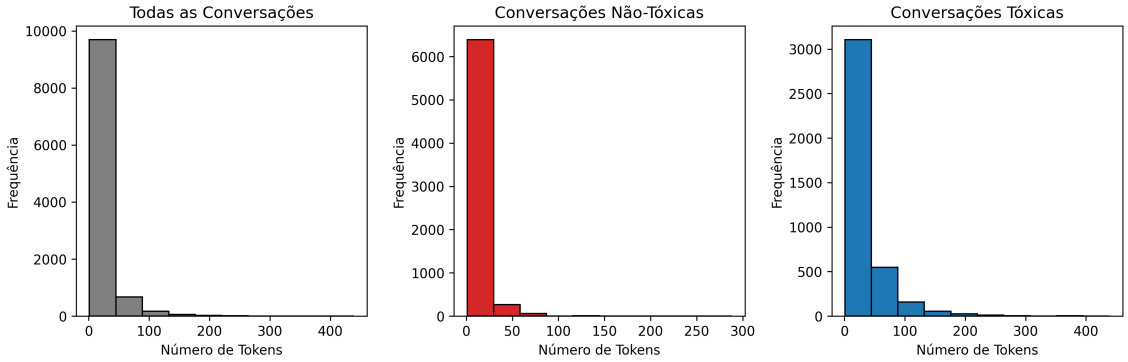


Figura 5.2: Histograma de frequência de número de *tokens* por tipo de conversação

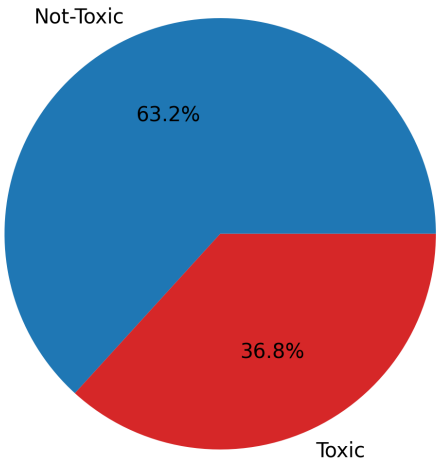


Figura 5.1: Distribuição de classes na base de dados

Na figura 5.2, podemos ver os histogramas com a distribuição de *tokens* na base de dados. Nota-se que a grande maioria das conversações é relativamente curta, mas conversações tóxicas tendem a ter um maior número de *tokens*, conforme corroborado pela figura 5.3, onde há um aumento relativo de $81,2\%$ no tamanho dos médio dos *tokens* (de $17,2$ para $31,5$) em relação a média geral, e de 250% (de 9 para $31,5$) em relação as conversações não tóxicas.

Na próxima seção discutiremos o processo de implementação da tarefa de classificação, com a definição do algoritmo e as métricas de avaliação escolhidas.

5.5 Execução

Nesta etapa do projeto, foram desenvolvidos os *scripts* necessários para a execução dos modelos, utilizando principalmente as bibliotecas *Transformers* e *SageMaker*. A biblioteca *Transformers* da *Hugging Face* foi empregada para facilitar a utiliza-

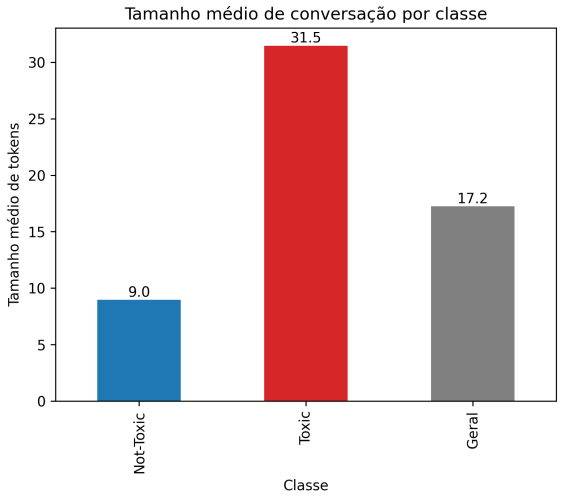


Figura 5.3: Tamanho Médio de Conversação por classe

ção de modelos de linguagem pré-treinados, permitindo uma rápida prototipagem e adaptação dos modelos às necessidades específicas da tarefa. Já o *SageMaker* da AWS foi utilizado para gerenciar e escalar os experimentos de *machine learning*, proporcionando a infraestrutura necessária.

Os *scripts* utilizados foram desenvolvidos para coletar os dados necessários para as avaliações propostas durante a execução dos modelos em cada *prompt* definido, incluindo o tempo de execução e o número de *tokens* de entrada e saída. Essas informações são usadas para avaliar o desempenho dos modelos e serão utilizadas posteriormente nas análises de performance e custo. Os códigos estão disponíveis no anexo C.

Métricas de Avaliação

Para a avaliação dos modelos implementados, foram utilizadas as métricas clássicas de classificação, a saber: *F1-score*, acurácia, precisão e *recall*. Estas métricas são amplamente reconhecidas e aceitas por sua capacidade de fornecer uma avaliação abrangente do desempenho dos modelos de classificação.

Essas métricas fornecem os dados necessários para uma análise detalhada do desempenho dos modelos, além de assegurar que os modelos executados atendam aos objetivos definidos. A coleta de dados e o uso dessas métricas foram fundamentais para a etapa subsequente de avaliação dos modelos, conforme descrito na seção 4.1.9.

Dado os requisitos do negócio, o sistema deve ser economicamente viável. Isso implica na construção de um *framework* experimental capaz de metrificar e considerar o custo-benefício e a viabilidade técnica como os principais pilares da pesquisa. Evitando esforços como *fine-tuning* ou uso de modelos muito custosos, caso possível.

Acurácia

É uma métrica básica e amplamente utilizada para avaliar modelos de classificação. Ela é definida como a proporção de previsões corretas (verdadeiros positivos e verdadeiros negativos) em relação ao número total de previsões. Formalmente, é dada por:

$$\text{Acc} = \frac{TP + TN}{TP + TN + FP + FN}, \tag{5.1}$$

onde:

- TP são os verdadeiros positivos,
- TN são os verdadeiros negativos,
- FP são os falsos positivos,
- FN são os falsos negativos.

Embora a acurácia forneça uma visão geral da performance do modelo, ela pode ser enganosa em bases de dados desbalanceadas, onde uma classe é muito mais prevalente que a outra. No contexto de identificação de toxicidade, se a maioria das interações forem não-tóxicas, um modelo que prevê todas as entradas como não-tóxicas pode ter alta acurácia, mas será ineficaz na detecção de comentários tóxicos.

Precision e Recall

Mede a proporção de previsões positivas corretas em relação ao total de previsões positivas feitas pelo modelo. É definida como:

$$\text{P} = \frac{TP}{TP + FP}, \tag{5.2}$$

onde TP são os verdadeiros positivos e FP são os falsos positivos.

Recall (ou Revocação) mede a proporção de verdadeiros positivos em relação ao total de casos positivos reais. É definida como:

$$\text{R} = \frac{TP}{TP + FN}, \tag{5.3}$$

onde FN são os falsos negativos. No contexto de identificação de toxicidade, um alto *recall* garante que a maioria dos casos tóxicos sejam identificados, enquanto uma alta precisão garante que a maioria das previsões de toxicidade sejam corretas.

F1-Score

O *F1-Score* é a média harmônica entre precisão e *recall*, proporcionando uma única métrica que equilibra ambos os aspectos:

$$F1 = 2 \cdot \frac{P \cdot R}{P + R}. \tag{5.4}$$

O F1-Score é particularmente útil quando há um desbalanceamento entre as classes, uma situação comum na identificação de toxicidade, onde os exemplos de comportamento tóxico podem ser muito menos frequentes do que os exemplos de comportamento não tóxico.

5.6 Seleção de Modelos

Devido ao grande interesse no campo e surgimento constante de novos modelos, a seleção de modelos relevantes torna-se uma tarefa não-trivial. Primeiramente, levantamos os recursos disponíveis para esta etapa, considerando que nosso maior custo, o de infraestrutura, se torna também o nosso maior cuidado na seleção, visto que modelos com maior número de parâmetros possuem também necessidades maiores de hardware.

Em seguida, estabelecemos as restrições, especialmente quanto ao tamanho dos modelos. As amplas exigências de *hardware* levaram ao desenvolvimento dos SLM, onde se busca condensar a performance de LLMs em modelos relativamente pequenos, normalmente até 10 bilhões de parâmetros, *e.g Gemma-2B e 8B* (GOOGLE, 2024) e *PHI-3* (LEPAGNOL *et al.*, 2024). Com isso, os resultados de desempenho têm melhorado consideravelmente nesse tipo de modelo (NAVEED *et al.*, 2024).

Considerando-se que os créditos da pesquisa estão disponíveis somente na AWS e seu uso será feito através da plataforma *AWS SageMaker*, nos limitamos a modelos disponíveis nesta, obtida diretamente do *Amazon SageMaker Python SDK*[6]. Em maio de 2024, a plataforma disponibilizava um total de 832 modelos, com aplicações variando de classificação de imagens a LLMs especializadas em geração de código. Aplicamos então, como restrição, a seleção de modelos até dez bilhões de parâmetros que estejam disponíveis na plataforma e possam ser utilizados na tarefa de classificação.

Posteriormente a esta filtragem, dada a natureza generalista dos LLMs e o grande número de candidatos, adicionamos uma filtragem para modelos do tipo *instruct*, pois são treinados para seguir instruções e demonstram desempenho superior aos modelos base em tarefas específicas (OUYANG *et al.*, 2022). Com isto, a lista final possuía 25 candidatos, onde aplicamos mais um filtro: estar apto a *fine-tuning*

[6]Disponível em `https://sagemaker.readthedocs.io/`

através da plataforma, reduzindo a lista a seis candidatos. O modelo *Mistral-7B-Instruct-v0.3* foi adicionado devido a sua relevância e popularidade, apesar de inapto a *fine-tuning* através da plataforma no tempo desta pesquisa.

Na tabela 5.9 podemos ver a lista de modelos candidatos e sua data de lançamento. A data de lançamento foi adicionada devido aos grandes avanços na performance dos modelos em um curto período de tempo, com melhorias significativas entre revisões modelos com número próximo de parâmetros, *e.g: Llama-2-7b* e *Llama-3-8b*.

Tabela 5.9: Modelos candidatos e sua data de lançamento

Família	**Modelos**	**Lançamento**
Gemma	*Gemma-2B-Instruct*, *Gemma-7B-Instruct*	02/2024
Llama-3	*Llama3-8b-Instruct*	04/2024
Mistral	*Mistral-7B-Instruct-v0.3*	10/2023
Falcon	*Falcon-7b-Instruct*	06/2023

Apesar de fora da lista de candidatos, os modelos *Falcon-40B-Instruct*, *Mixtral-7x8B-Instruct-v0.1* e *Llama-3-70B-Instruct* foram incluídos nas comparações de desempenho devido ao maior número de parâmetros, todavia, devido a limitações de custo e tempo, foram analisados apenas os *prompts* em configuração *Zero Shot* para uso como linha base em algumas análises de desempenho.

Os modelos *GPT-3.5-turbo-0125*, *GPT-4-0125-preview*, *GPT-4o* também foram incluídos seguindo o mesmo racional. Outra prerrogativa para adição dos modelos GPT é a sua facilidade no acesso a modelos com número de parâmetros muito superior aos modelos candidatos de código aberto, conforme tabela 5.10.

Tabela 5.10: Modelos adicionais, seu número de parâmetros e data de lançamento

Modelos	**Número de Parâmetros**	**Lançamento**
Falcon-40B-Instruct	*40 bilhões*	06/2023
GPT-3.5-turbo-0125	*180 bilhões*	01/2024
GPT-4-0125-preview	*1,76 trilhões*	01/2024
GPT-4omni	*1,76 trilhões*	05/2024
Mixtral-7x8B-Instruct-v0.1	*47 bilhões*	12/2023
Llama-3-70B-Instruct	*70 bilhões*	04/2024

Na próxima seção apresentaremos as estratégias de *prompt* selecionadas, conjuntamente com os *prompts* definidos para a experimentação.

5.7 Elaboração de *Prompts*

Nesta seção, detalhamos a seleção das estratégias de *prompt* utilizadas nos experimentos para a classificação de toxicidade em *chats* de Dota 2. Dado o abundante número de estratégias, conforme demonstrado na seção 3.4, optou-se por utilizar as estratégias *Zero-Shot* e *Few-Shot*, cada uma com e sem o uso de *Chain-of-Thought*.

A abordagem de *Chain-of-Thought* foi incluída para investigar se um raciocínio passo a passo pode melhorar a qualidade da classificação, devido ao seu potencial de melhorar o raciocínio dos modelos de linguagem ao decompor problemas complexos em etapas menores e mais gerenciáveis. WEI *et al.* (2023); KOJIMA *et al.* (2023) demonstraram que o *Chain of Thought Prompting* pode aumentar significativamente a capacidade dos LLMs em tarefas de uso final, indicando desempenho superior a técnicas basais como *Zero Shot* e *Few Shot*.

Outro elemento considerado na pesquisa foi o impacto das definições das classes da tarefa no *prompt*. Para cada uma dessas estratégias, testamos variações de *prompts* que incluem ou não a definição das classes da tarefa de classificação: determinar se o conteúdo é tóxico ou não tóxico. A inclusão das definições visa fornecer uma melhor orientação ao modelo, potencialmente melhorando a qualidade da classificação. As variações de *prompt* consideradas foram:

- **Nenhuma Definição**: não fornecem qualquer definição sobre o que caracteriza um conteúdo tóxico ou não tóxico.
- **Definição de Tóxico (T)**: incluem uma definição sobre o que caracteriza um conteúdo tóxico.
- **Definição de Não Tóxico (NT)**: incluem uma definição sobre o que caracteriza um conteúdo não tóxico.
- **Definição de Ambos (A)**: incluem definições tanto do que caracteriza um conteúdo tóxico quanto do que caracteriza um conteúdo não tóxico.

Considerando o número de estratégias e variações escolhidos, cada modelo foi testado num total de dezesseis diferentes *prompts*. O impacto causado pelos *prompts* é muito relevante no desempenho e o número de variações dentro de um mesmo *prompt* é imensa, podemos descrever toxicidade de múltiplas formas, cada variação gerando mais cenários para análise, tornando a busca pelo *prompt* ótimo um cenário improvável de ser atingido. É importante destacar que essa análise não visa encontrar o *prompt* ótimo, entretanto, os resultados destes experimentos permitirão uma melhor compreensão da eficácia dessas estratégias e o impacto da definição das classes na tarefa de classificação.

Na tabela 5.11, temos uma lista das estratégias e suas definições de acordo com o contexto adicional das classes adicionado nos *prompts*.

Tabela 5.11: Abreviações dos *prompts* testados em cada abordagem

Estratégia	**Definição**			
	Nenhuma	**Tóxico**	**Não Tóxico**	**Ambos**
Zero-Shot	ZS	ZS-T	ZS-NT	ZS-A
Few-Shot	FS	FS-T	FS-NT	FS-A
Zero-Shot CoT	ZSCoT	ZSCoT-T	ZSCoT-NT	ZSCoT-A
Few-Shot CoT	FSCoT	FSCoT-T	FSCoT-NT	FSCoT-A

Na próxima seção discutiremos as configurações para execuções dos *prompts* em cada modelo. A lista completa de *prompts* encontra-se disponível no anexo B.

5.8 Execução de Modelos

Durante a execução dos modelos, foram executadas as adaptações necessárias para os modelos usando os *prompts* propostos. Cada modelo possui requisitos diferentes de hardware, e cada família de modelos utiliza um formato diferente de instrução, que deve ser utilizado em conjunto com os *prompts* propostos garantindo a compreensão do modelo das entradas fornecidas. Quanto a configuração dos *prompts*, todos foram executados com temperatura próxima a zero, buscando respostas determinísticas na saída do modelo, e com número máximo de *tokens* de saída em 256.

O uso de LLMs implica em exigências de hardware muito superiores a hardware consumidor, devido ao seu alto consumo de *VRAM* de *GPUs* para tarefas como inferência ou treinamento. Modelos grandes, como o *Llama3-70b* utilizam cerca de 100GB de *VRAM*, o equivalente a seis *GPUs* de 24GB[7] aproximadamente.

Considerando a lista de modelos candidatos apresentados na tabela 5.5 e sua natureza de código-aberto, utilizamos os modelos prontamente disponíveis na plataforma *HuggingFaces*[8]. Utilizamos as máquinas mínimas recomendadas dentro do *AWS SageMaker*[9], especificadas na tabela 5.12. Foram utilizadas *instâncias G5*[10], a última geração de instâncias da *Amazon* baseadas em GPUs *NVIDIA*[11] em todos os experimentos. Existe uma diferença significativa no custo dos modelos com maior número de parâmetros.

[7] VRAM de uma *GeForce RTX 4090*, topo de linha para consumidores em julho de 2024
[8] `https://huggingface.co/`
[9] `https://aws.amazon.com/pm/sagemaker/`
[10] `https://aws.amazon.com/pt/ec2/instance-types/g5/`
[11] `https://nvidia.com`

Tabela 5.12: *Instâncias do tipo G5 usadas nos modelos candidatos*

Modelos	**Instância**	**Custo (USD/h)**
Gemma-2B	g5.xlarge	$1.006
Mistral-7B-Instruct-v0.2, *Gemma-8B-Instruct*, *Falcon-7b-Instruct*, *Llama-3-8B-Instruct*	*g5.2xlarge*	$1.212
Falcon-40B-Instruct, *Mixtral-8x7B-Instruct-v0.1*, *Llama-3-70B-Instruct*	*g5.48xlarge*	$16.288

Após o ajuste dos *prompts*, o *fine-tuning* pode ser aplicado ou não nesses modelos, entretanto, para este trabalho, devido a seu foco na economicidade, restrições de tempo e o cenário proposto de uma desenvolvedora, não realizamos esse procedimento em nenhum modelo, usando de técnicas de engenharia de *prompt* nos modelos base devido a sua flexibilidade.

Na próxima seção, executamos uma comparação detalhada de desempenho dos modelos candidatos na tarefa de classificação. Iniciando com uma análise de desempenho, utilizando as métricas de avaliação definidas anteriormente — *F1-score*, acurácia, precisão e *recall*. O código para predição pode ser encontrado no apêndice C.

5.9 Análise de Desempenho de Modelos

Nesta seção serão apresentados os resultados de desempenho obtidos nos experimentos, com o objetivo de analisar o desempenho dos modelos candidatos na tarefa de classificação de toxicidade. Essas análises são feitas através do uso das métricas definidas na seção 5.5. Além disso, aplicamos diferentes técnicas de *Prompt Engineering*, totalizando dezesseis diferentes *prompts* por modelo candidato, conforme seção 5.7. Os modelos adicionais foram analisados apenas no *prompt zero-shot-none*. As alucinações nos resultados como erros durante os cálculos das métricas definidas, o que será discutido com maiores detalhes na seção 5.9.2.

Durante a discussão dos resultados, usaremos notações e termos para os *prompts* analisados. Uma *estratégia de prompt* se refere a uma das estratégias de *Prompt Engineering* utilizadas, sendo estas *Zero-Shot, Few-Shot, CoT-Zero-shot e CoT-Few-Shot*. Uma *variação* refere-se aos diferentes *prompts* de cada estratégia, de acordo com as definições da classe concatenadas ao *prompt*, a saber *Toxic, Not-Toxic, Both, None*. Um *prompt* é a totalidade do *prompt*, com estratégia e definição, *e.g zero-shot-none*, *cot-zero-shot-both*. O uso da palavra *modelo* refere-se a um LLM, e *família de modelos* refere-se a modelos que possuem mesmo nome e arquitetura de treinamento, entretanto, possuem diferente número de parâmetros *e.g Llama-3-8B* e *Llama-3-70B*.

Os experimentos foram conduzidos em dois momentos: uma etapa para identificar o melhor desempenho dentre as variações de *prompt*, avaliando os modelos em custo e desempenho e uma etapa de análise do modelo escolhido. Na primeira etapa, utilizamos os 30% da base conforme definido na seção 5.4, dado o alto custo computacional envolvido. Na etapa de análise do modelo escolhido, utilizamos a base em sua completude.

Um fator crucial na análise dos resultados é a taxa de alucinação dos modelos. Todos os LLMs são suscetíveis a alucinações, que representam respostas factualmente incorretas do modelo a um *prompt*, e cada modelo possui uma tendência diferente a alucinações. Na análise dos resultados demonstrados abaixo, consideramos que as alucinações dos modelos foram consideradas como erros. Discutiremos a taxa de alucinação na seção 5.9.2.

O modelo *Falcon-40B-Instruct* foi omitido dos resultados, devido a um número de alucinações próximos a totalidade das classificações, causado por mecanismos de segurança do próprio modelo, uma das maiores preocupações quanto ao desenvolvimento de modelos éticos e seguros, para prevenir usos inapropriados dos mesmos, e até mesmo atender a regulamentações (DONG *et al.*, 2024). Modelos como o *Llama-3* (TOUVRON *et al.*, 2023) e *Gemma* (GOOGLE, 2024) possuem mecanismos de proteção filtrando a entrada e saída dos modelos. A resposta *"I cannot classify the given Dota 2 chat message as Toxic or Not-Toxic as it is incomplete and lacks context"* mostra um exemplo de saída desse modelo. Esse comportamento foi observado apenas nesse modelo.

O modelo *Mixtral-8x7b-Instruct-v0.1* também foi omitido pois estava indisponível na plataforma *AWS SageMaker* devido a erros técnicos no tempo dessa pesquisa.

5.9.1 Exploração de *Prompts*

Aplicamos as mesmas estratégias de *prompt* e variações em todos os modelos candidatos, com exceção dos modelos adicionais. Considerando-se as diferenças no conteúdo usado no pré-treinamento de cada modelo, seguimos o racional que diferentes modelos reagirão de forma diferente aos *prompts*. É importante notar que, dentre os modelos analisados, o modelo *Gemma-2B-Instruct* possui um número bem inferior de parâmetros em relação aos modelos candidatos e os modelos adicionais possuem número de parâmetros vastamente superior, especialmente no caso dos modelos da família *GPT*.

Na figura 5.4, podemos ver que o *prompt Chain-of-Thought* em *Zero Shot* e sem nenhuma definição como contexto ocupando a primeira e terceira colocações, respectivamente, seguido dos modelo *GPT-4o* e *Llama-3-70B-Instruct* em configuração *Zero Shot* e sem definição. A diferença dos resultados e variação nos melhores

prompts por modelo confirma a hipótese que cada modelo reage de maneira diferente aos *prompts*.

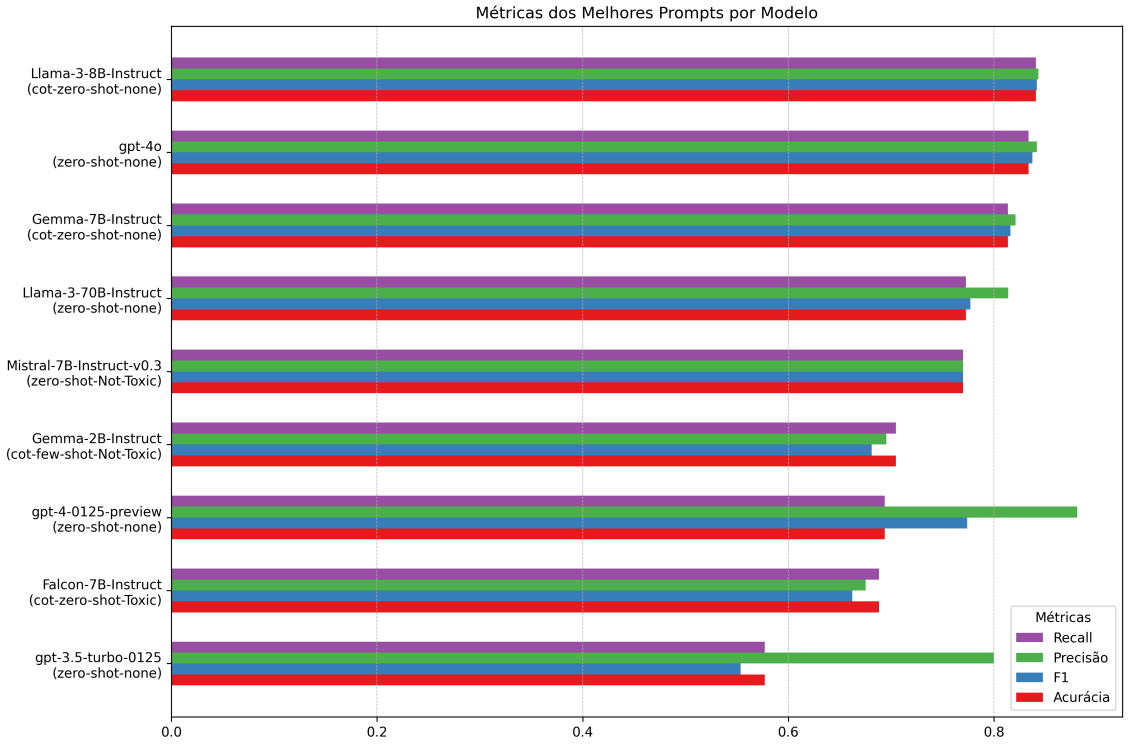


Figura 5.4: Métricas dos melhores *prompts* por modelo

O modelo *Falcon-7B-Instruct* teve o pior desempenho dos candidatos, com resultados inferiores ao *Gemma-2B-Instruct*, que possui apenas 25% dos parâmetros em comparação. Uma hipótese plausível seria que o campo das SLM têm evoluído muito rapidamente, com novos modelos atingindo desempenho superior com número menor de parâmetros, em modelos mais robustos. Essas melhoras expressivas têm sido atingidas através de otimizações em arquitetura, conjunto de dados de treinamento, dentre outros (SAHOO *et al.*, 2024). Na tabela 5.9 podemos ver uma distância temporal de 8 meses entre o lançamento dos modelos *Falcon* e *Gemma* (06/23 e 02/24, respectivamente).

A figura 5.5 mostra o desempenho dos modelos para cada variação de *prompt*. Notavelmente, as estratégias *Few Shot* e *CoT-Few-Shot* tiveram os piores resultados num geral, indicando que o uso de exemplos puramente pode degradar o desempenho nessa categoria de modelo de até dez bilhões de parâmetros, exceto no modelo *Gemma-2B-Instruct* onde o prompt *CoT-Few-Shot-Not-Toxic* teve seu melhor desempenho.

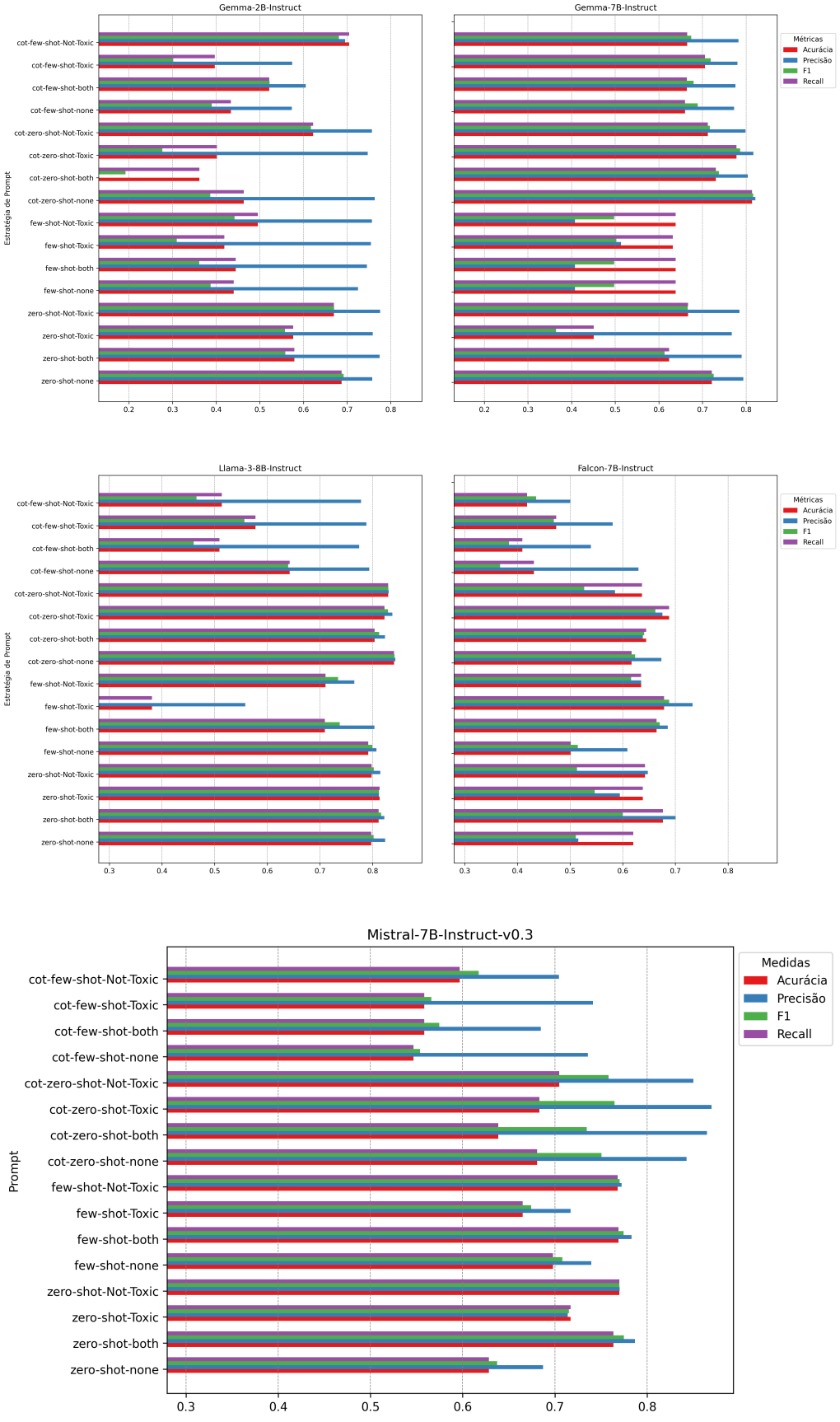


Figura 5.5: Desempenho dos *prompts* em cada modelo

A estratégia *CoT-Zero-Shot* e suas variações apresentou resultados consistentes em todos os modelos, exceto no *Gemma-2B-Instruct*. É também importante observar o desempenho do *prompt Zero-Shot-None*, devido ao baixo custo de planejamento e desempenho próximo às estratégias com melhor desempenho, exceto nos modelos *Falcon-7B-Instruct* e *Mistral-7B-Instruct-v0.3*. Os *prompts* incluindo definições do conceito de toxicidade impactaram o modelo negativamente em algumas estratégias, com a variação *none* tendo o melhor desempenho majoritariamente. Uma hipótese plausível é que os modelos possivelmente já possuem uma boa definição do que é toxicidade advindo do seu pré-treinamento, tornando a adição de definições positiva em poucos casos.

Segundo WEI *et al.* (2023), a estratégia *Chain-of-Thought* visa elicitar a capacidade *reasoning* do modelo, e sua eficácia aumenta em modelos com maior número de parâmetros, o que pode ser corroborado no gráfico 5.6, onde dois modelos de uma mesma família e com diferentes números de parâmetros possuem desempenho consideravelmente diferentes nesta estratégia. Nos outros modelos, os *prompts* com estratégia desse tipo exibiram desempenho superior de maneira geral. No *prompt CoT-Zero-Shot-None*, por exemplo, temos uma melhora drástica no aumento do F1-Score com 107% de melhoria aproximadamente (de 0.39 para 0.80).

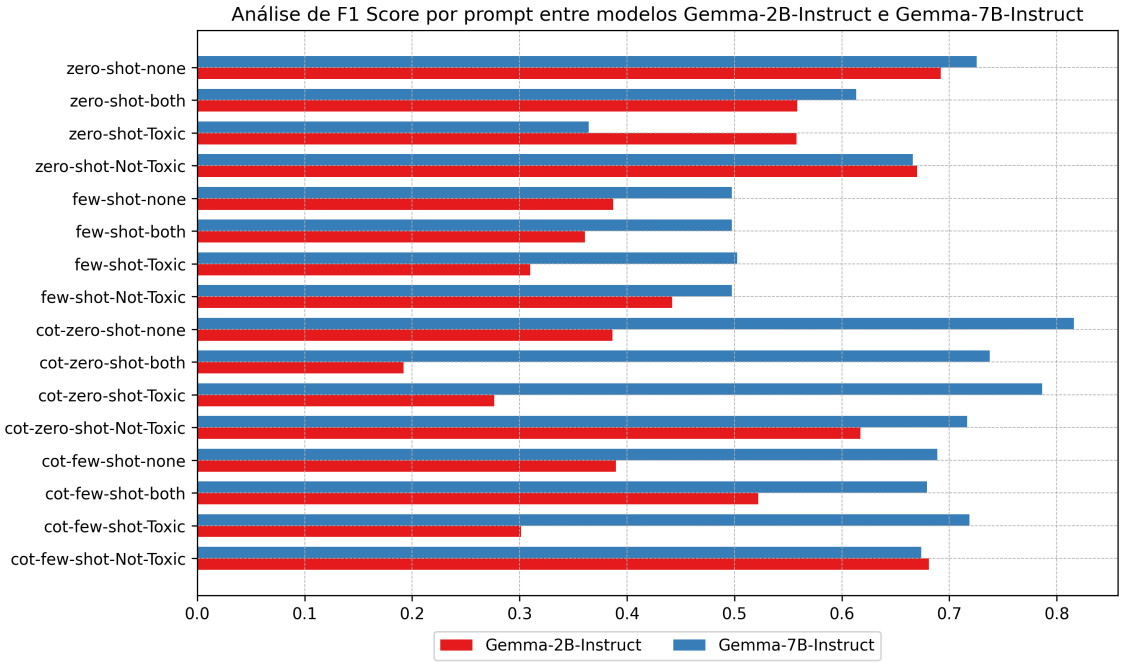


Figura 5.6: Comparação do F1 Score por Estratégia de Prompt

Modelos Adicionais

A tabela 5.13 mostra o desempenho dos modelos candidatos e adicionais no *prompt Zero Shot None*, usado como linha base de comparação. Podemos observar que o

número de parâmetros não é determinante no desempenho dessa tarefa, com o modelo *gpt-3.5-turbo-0125* apresentando desempenho inferior ao modelo *Gemma-2B-Instruct*, reforçando a hipótese das melhorias substanciais dos SLM recentemente.

Tabela 5.13: *Métricas dos modelos em prompt Zero-Shot None.* Os modelos com melhores resultados estão em destaque

Modelo	**Parâmetros**	**F1**	**Precisão**	**Recall**	**Acurácia**
Falcon-7B-Instruct	7B	0,51	0,52	0,62	0,62
Gemma-2B-Instruct	2B	0,69	0,76	0,69	0,69
Gemma-7B-Instruct	7B	0,73	0,79	0,72	0,72
Llama-3-70B-Instruct	70B	0,78	0,81	0,77	0,77
Llama-3-8B-Instruct	8B	0,80	0,82	0,80	0,80
Mistral-7B-Instruct-v0.3	7B	0,64	0,69	0,63	0,63
gpt-3.5-turbo-0125	175B	0,55	0,80	0,58	0,58
gpt-4-0125-preview	1,76T	0,77	0,88	0,69	0,69
gpt-4o	1,76T	0,84	0,84	0,83	0,83

O modelo com melhor desempenho, *gpt-4o* possui desempenho semelhante aos modelos da família *Llama-3*, com destaque no modelo *Llama-3-8B-Instruct*, apresentando resultados marginalmente superiores ao modelo *Llama-3-70B-Instruct*, significativamente maior. A proximidade nos resultados desses modelos menores, em comparação com um modelo 220 vezes maior em número de parâmetros (8 bilhões para 1,76 trilhão), exibe também um comportamento de retornos decrescentes em *context learning* (BROWN *et al.*, 2020), a partir de um certo volume de parâmetros na tarefa avaliada.

5.9.2 Taxa de Alucinação

Outro fator importante a ser considerado na análise de desempenho dos modelos é sua tendência a alucinações. Esse comportamento é particularmente prejudicial na tarefa de classificação, visto que cria uma nova classe com as alucinações. Seguindo o racional que os resultados do modelo são afetados pelo *prompt*, analisamos os diferentes *prompts* de acordo com a taxa de alucinações do modelo. Na figura 5.7, podemos ver as o impacto de cada *prompt* na taxa de alucinação dos modelos.

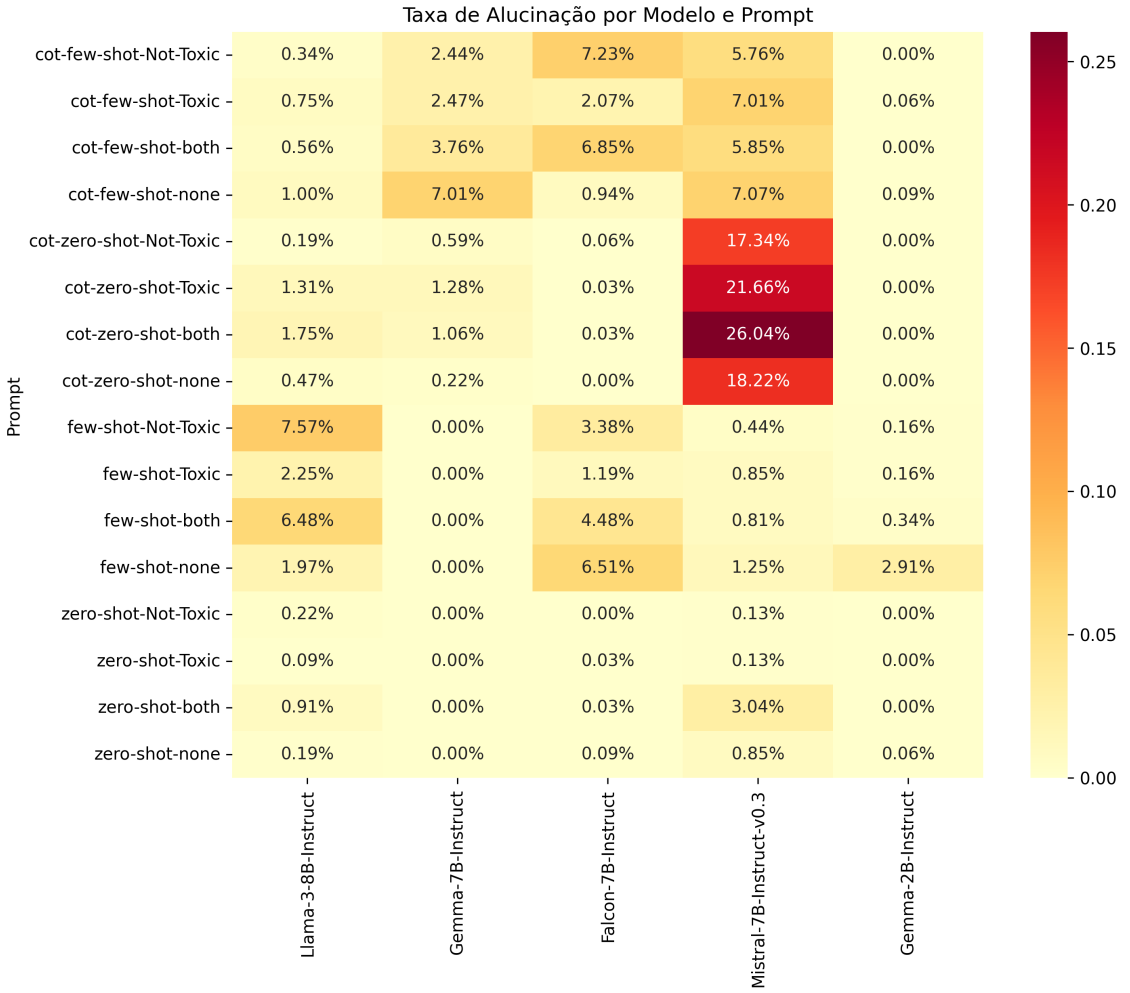


Figura 5.7: Mapa de calor de taxa de alucinação por modelo e *prompt*

Durante a experimentação, o modelo *Mistral-7B-Instruct-v0.3* criou uma nova classe de classificação, a classe *neutra*, que, para fins de análise, foi convertida para a classe *não tóxica*, logo, a taxa de alucinação desse modelo é calculada após essa conversão. O modelo sofreu uma degradação no seu desempenho nas estratégias do tipo *CoT*, com taxas de alucinação elevadas. Os *prompts* *CoT-Few-Shot* também tiveram desempenho inferior em relação às outras duas estratégias nesse modelo, demonstrando uma baixa compatibilidade com *prompts* do tipo *CoT* nesse modelo.

Na tabela 5.14, podemos ver os *prompts* com as maiores taxas de alucinação por modelo. Dentre os modelos candidatos, com exceção do modelo *Mistral-8B-Instruct-v0.3*, os piores *prompts* em cada modelo alguma variação de *prompt* em *Few Shot*. O modelo com a menor taxa de alucinações foi o *Gemma-2B-Instruct*.

Na figura 5.8, podemos ver as matrizes de confusão do modelo *Mistral-7B-Instruct-v0.3*, *prompts* com maior taxa de alucinação dentre todos os modelos. Nas figuras 5.8a, 5.8b e 5.8c, mostram uma tendência do modelo em alucinar mais em amostras não tóxicas, com taxas de alucinações superiores nessas variações, e possuindo também a maior taxa de verdadeiros positivos. Na figura 5.8d, onde há uma

Tabela 5.14: *Prompts com maiores taxas de alucinação dos modelos. Os modelos adicionais (destacados em cinza) foram analisados apenas no prompt zero-shot-none*

Modelo	**Prompt**	**Taxa Alucinação (%)**	**Alucinações**
Mistral-7B-Instruct-v0.3	cot-zero-shot-both	26.04	832
gpt-4-0125-preview	zero-shot-none	20.22	646
Llama-3-8B-Instruct	few-shot-Not-Toxic	7.57	242
Falcon-7B-Instruct	cot-few-shot-Not-Toxic	7.23	231
Gemma-7B-Instruct	cot-few-shot-none	7.01	224
Gemma-2B-Instruct	few-shot-none	2.91	93
gpt-4o	zero-shot-none	1.06	34
gpt-3.5-turbo-0125	zero-shot-none	1.00	32
Llama-3-70B-Instruct	zero-shot-none	0.00	0

definição clara de *não tóxico*, essa tendência é invertida, com o modelo passando a ter mais alucinações em amostras tóxicas, seguido do maior número de verdadeiros negativos dentre o grupo. O *prompt* que define ambos, *cot-zero-shot-both*, possui o pior desempenho, indicando que as definições têm um impacto significante para essa estratégia.

Seguido do modelo *Mistral-7B-Instruct-v0.3*, o modelo *GPT-4-0125-preview* também apresentou uma taxa de alucinação elevada no *prompt zero-shot-none*. Na figura 5.9, podemos ver que o modelo teve um maior número de alucinações em amostras não tóxicas. Esse modelo possui uma característica singular, com a maior precisão dentre todos os modelos analisados, com a menor taxa de *falsos positivos*.

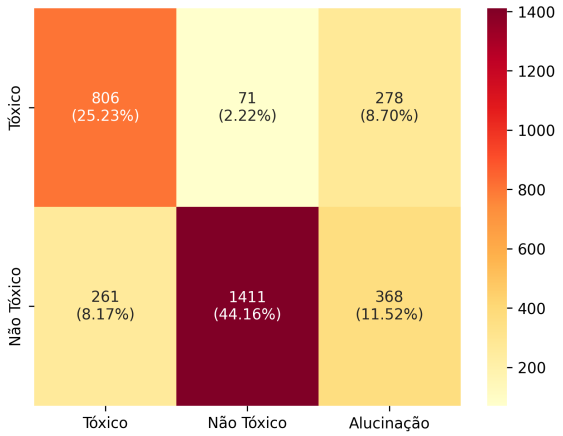
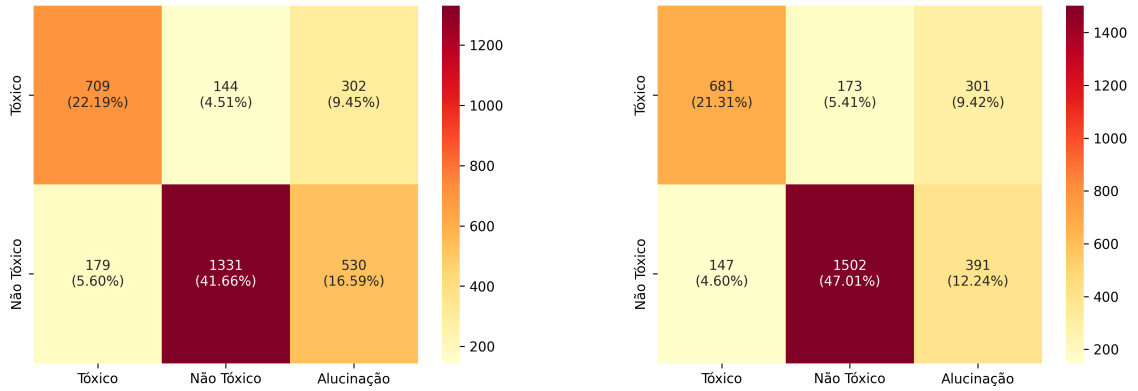


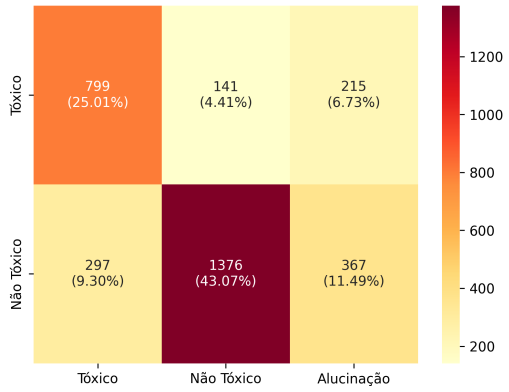
Figura 5.9: Matriz de confusão do modelo *GPT-4-0125-preview* em *prompt zero-shot-none*

Considerando os resultados de desempenho apresentados, concluímos que os modelos *Llama-3-8B-Instruct*, *GPT-4o* e *Gemma-7B-Instruct* obtiveram resultados próximos e, numa análise estritamente de desempenho, seriam os candidatos a modelo final escolhido, nessa ordem, nos *prompts* definidos na figura 5.4. Contudo, a escolha de um modelo para o problema da pesquisa também precisa levar em conta a economicidade desses modelos.

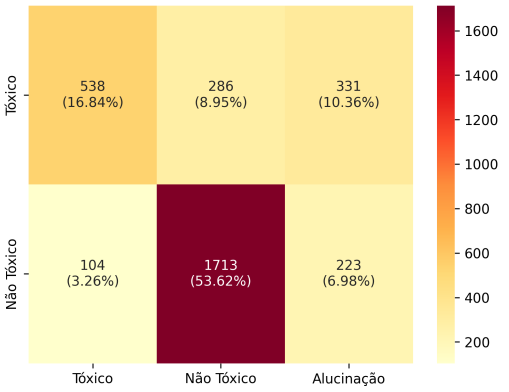


(a) *cot-zero-shot-both*

(b) *cot-zero-shot-toxic*



(c) *cot-zero-shot-none*



(d) *cot-zero-shot-not-toxic*

Figura 5.8: Matrizes de confusão dos *prompts* com maior taxa de alucinação no modelo *Mistral-7B-Instruct-v0.3*

Na próxima seção, faremos uma análise do custo dos modelos, incluindo os adicionais, relacionando seu custo com o desempenho de cada *prompt*, considerando um denominador comum para os diferentes custos envolvidos ao utilizar os modelos em diferentes plataformas, como *AWS SageMaker* e *OpenAI*.

5.10 Análise de Custo

Além da avaliação de desempenho, é crucial analisar os custos associados à implementação e execução dos modelos. Existem fatores prevalentes ao analisar custos de LLMs, de acordo com qual plataforma se está usando para consumo dos modelos. Neste trabalho, utilizamos duas plataformas: uma de *Cloud Computing* (AWS *SageMaker*) e através de APIs externas no caso dos modelos *GPT*.

Cada plataforma de consumo desses LLMs possui vantagens e desvantagens, conforme pode ser visto na tabela 5.15. O uso de APIs externas oferece vantagens significantes quanto a experimentação e prototipação, com escalabilidade, configuração inicial, facilidade de uso e manutenção garantidos como parte do serviço, comumente o efeito colateral nesse tipo de serviço é o preço mais elevado. As limitações

no uso dessas plataformas fica clara também quanto a customização, escalabilidade e principalmente privacidade, tópico que temestado em alta devido ao uso de dados sensíveis dos usuários nesses modelos.

Tabela 5.15: *Comparação entre o uso de LLMs através da plataforma de Cloud Computing SageMaker e APIs externas (como OpenAI).*

Aspecto	**AWS SageMaker**	**APIs Externas**
Configuração Inicial	Requer configuração e gerenciamento de infraestrutura na AWS.	Sem necessidade de configuração de infraestrutura; uso direto via API.
Custos	Pagamento por uso de instâncias EC2, armazenamento, e outros recursos AWS.	Pagamento por tokens processados; sem custos de infraestrutura.
Escalabilidade	Alta escalabilidade com controle sobre recursos de computação.	Escalabilidade gerida pelo provedor da API; limitado pela oferta do serviço.
Customização	Controle completo sobre o ambiente de execução e a possibilidade de customizar modelos.	Limitações na customização de modelos; dependência das configurações padrão da API.
Segurança e Privacidade	Alto controle sobre dados e segurança dentro do ambiente AWS.	Dependente das políticas de segurança do provedor da API; possível exposição de dados sensíveis.
Latência	Potencialmente menor se configurado adequadamente dentro da região AWS.	Depende da localização dos servidores do provedor da API e da rede de entrega.
Facilidade de Uso	Mais complexo devido à necessidade de gerenciar infraestrutura.	Fácil de usar com integração direta via API.
Manutenção	Requer manutenção contínua da infraestrutura e atualizações.	Sem manutenção de infraestrutura; provedor da API gerencia tudo.

5.10.1 Custo de Modelo de Linguagem

O uso de diferentes plataformas de consumo implica em formas diferentes de mensurar e calcular o custo. No caso da API externa utilizada através da OpenAI, o custo é calculado pelo número de *tokens* de entrada e de saída por milhão de *tokens*, por modelo. No caso de modelos na plataforma *AWS SageMaker*, utilizamos os custos associados, majoritariamente instâncias, para calcular o custo de cada *token*, de acordo com o tempo de processamento gasto pelo modelo para processar a base de testes.

Devido às diferenças em forma de cobrança de cada plataforma, buscamos uma métrica que pudesse possibilitar uma comparação de custos entre as duas formas de consumo. Escolhemos o custo por inferência como métrica, dado as métricas disponíveis, ou que podemos inferir. No caso da *OpenAI*, não é possível medir diretamente os custos relacionados a infraestrutura *e.g* custo das instâncias, tempo computacional e armazenamento. Ao utilizar a plataforma *AWS SageMaker*, não é possível medir o custo dos *tokens* de entrada e saída diretamente, visto que possuímos apenas os custos gerais, baseado primariamente no custo das instâncias utilizadas. A tabela 5.16 mostra as variáveis que podemos medir em cada plataforma e a notação que utilizaremos.

Tabela 5.16: *Métricas de custo mensuráveis em cada plataforma*

Variáveis	Notação	OpenAI	AWS SageMaker
Custo Tokens Entrada	CT_E	✓	✗
Custo Tokens Saída	CT_S	✓	✗
Custo Instâncias	C_I	✗	✓
Tempo Total Inferência	T_I	✓	✓
Tokens Entrada	T_E	✓	✓
Tokens Saída	T_S	✓	✓

Os *tokens* de entrada são os *tokens* do *prompt* enviados pelo usuário para o modelo. Isso inclui todos os dados de contexto, exemplos e todas as informações necessárias para completude do *prompt*. Os *tokens* de saída são os *tokens* que são retornados como resposta do modelo na tarefa de inferência. Os custos das instâncias refere-se ao custo das instâncias utilizadas para execução do modelo, de acordo com a seção 5.10.1. O tempo total de inferência refere-se ao tempo levado para classificar toda a base no *prompt* atual. Utilizaremos a notação C_T para definir o custo total da tarefa em toda a base de teste.

$$C_{T_{SageMaker}} = T_I \times C_I. \tag{5.5}$$

A equação 5.5 mostra o cálculo do custo total por um *prompt* no *AWS SageMaker*, onde desconsideramos custos menores, como *logs* e armazenamento, e utilizamos apenas o tempo de inferência total dividido pelo custo da instância, conforme tabela 5.12. O cálculo do custo médio por inferência do modelo, é demonstrada na equação 5.6, através da divisão do custo total, pelo total de amostras.

$$\text{CustoInferencia}_{SageMaker} = \frac{C_{T_{SageMaker}}}{\#\ \text{Amostras}}. \tag{5.6}$$

Na plataforma *OpenAI* utilizamos os custos de entrada e saída por milhão de *tokens* nos modelos analisados em Junho de 2024, conforme tabela 5.17. A equação 5.7

representa o custo total por *prompt* através da soma total dos *tokens* de entrada e saída, multiplicado pelo custo dos mesmos, em seguida a equação 5.8 representa o custo médio por inferência dividindo-se o custo total pelo total de amostras analisadas.

$$C_{T_{OpenAI}} = (\sum_{i=1}^{N} \mathrm{T}_{E_i} \times CT_E) + (\sum_{i=1}^{N} \mathrm{T}_{S_i} \times CT_S) \tag{5.7}$$

$$\mathrm{CustoInferencia}_{OpenAI} = \frac{C_{T_{OpenAI}}}{\#\ \mathrm{Amostras}} \tag{5.8}$$

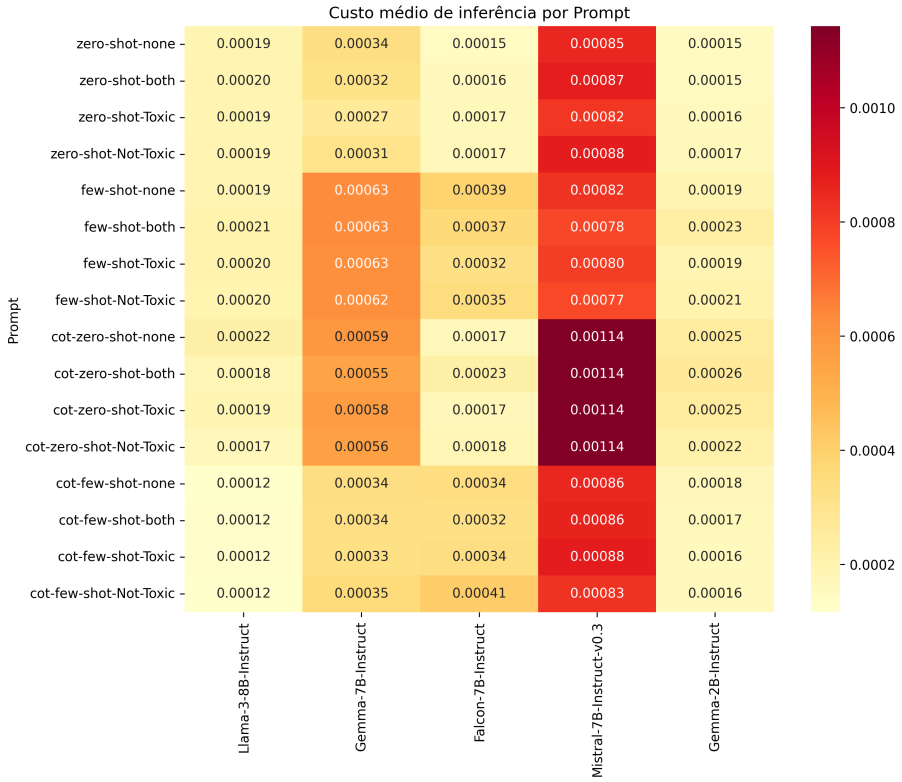
Tabela 5.17: Preços em dólar americano por milhão de tokens, para cada modelo da *OpenAI*

Modelo	**Entrada**	**Saída**
gpt-4o, *gpt-4o-05-13*	$5.00	$15.00
gpt-4-0125-preview	$10.00	$30.00
gpt-3.5-turbo-0125	$0.50	$1.50

5.10.2 Resultados

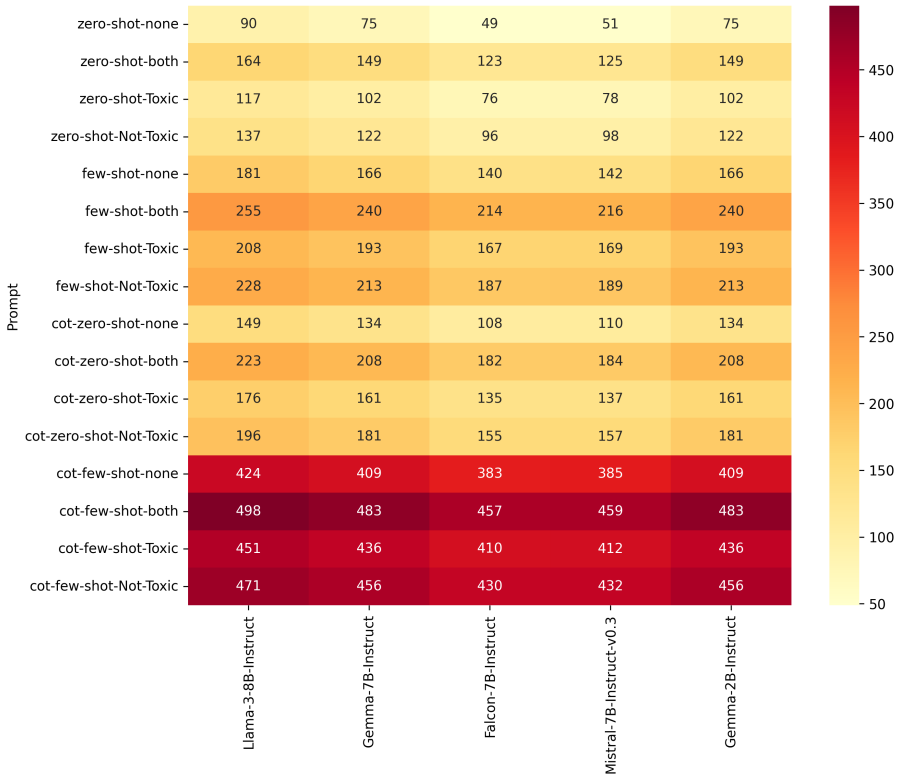
De acordo com as equações definidas na seção 5.10.1, calculamos os valores do custo médio por inferência em cada modelo. No gráfico 5.10, podemos ver que o modelo *Mistral-7B-Instruct-v0.3* teve o maior custo dentre os modelos, seguido do modelo *Gemma-7B-Instruct*. Os modelos *Llama-3-8B-Instruct* e *Gemma-2B-Instruct* obtiveram os menores custos dentre os modelos candidatos.

Figura 5.10: *Mapa de calor com o custo por inferência por prompt dos modelos candidatos*



Foi esperado uma diferença significativa nos custos de inferência devido a diferença no número de *tokens*, tanto de entrada como de saída, em cada *prompt*. Na figura 5.11, podemos ver os *tokens* de entrada e saída em cada *prompt*. Podemos ver a diferença entre os *tokens* de entrada em cada prompt na figura 5.11a, onde os *prompts* do tipo *cot-few-shot* possuem um número maior de *tokens* comparado aos demais. Apesar do *prompt* mais verboso, podemos ver na figura 5.11b, que não há uma relação direta entre o número de *tokens* de entrada e os de saída dentre os *prompts* analisados.

Apesar de não conclusivo, parece haver uma relação entre o custo e o número de *tokens* de saída de cada *prompt*, de acordo com as figuras 5.11b e 5.10, onde quanto maior o número de *tokens* de saída, maior o custo médio da inferência. Essa hipótese se apoia na própria estrutura da tarefa de inferência executada pelos LLMs, onde cada *token* de saída é gerado. A diferenças nos valores de *tokens* de entrada e saída exibidas na tabela 5.17 também reforçam o racional no custo de processamento dos *tokens* em cada uma das etapas da tarefa de inferência.



(a) *Tokens* de Entrada

(b) *Tokens* de Saída

Figura 5.11: *Mapas de calor com número de tokens de entrada e saída por prompt*

Na figura 5.12, relacionamos a *acurácia* dos *prompts* e seu custo médio de inferência. O modelo *Llama-70B-Instruct* foi considerado um *outlier* e omitido para

melhor visualização, devido ao seu custo vastamente superior aos demais, conforme tabela 5.18.

Figura 5.12: *Gráfico relacionando acurácia com custo médio por inferência. Cada item representa um prompt. Resultados no quadrante superior esquerdo são melhores. O modelo Llama-70B-Instruct foi considerado um outlier e omitido para melhor visualização, devido ao seu custo vastamente superior aos demais, conforme tabela 5.18.*

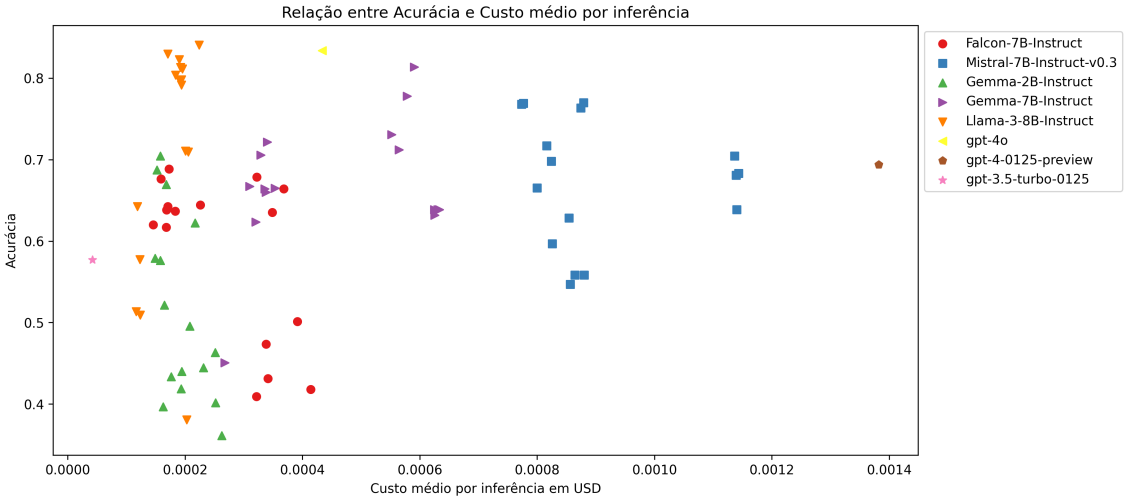


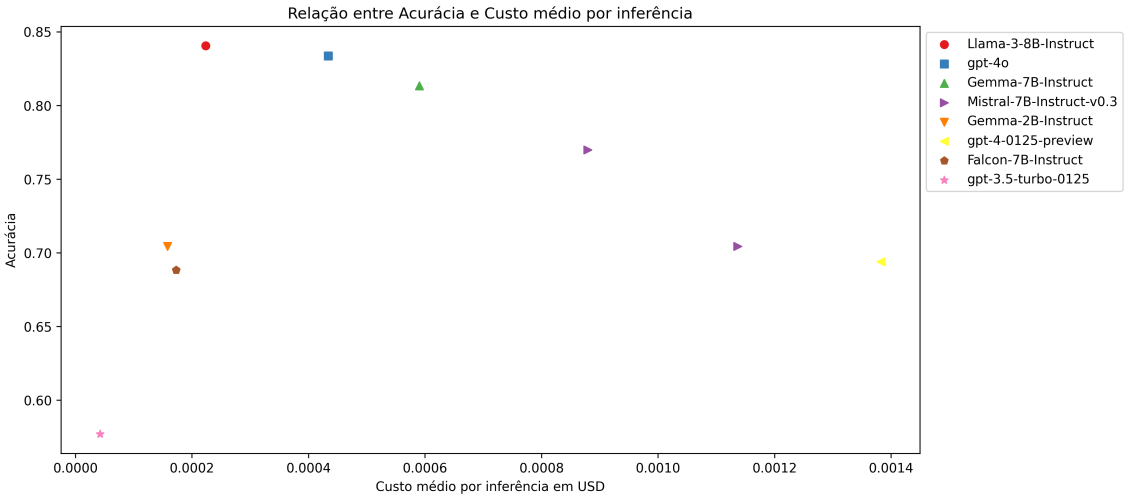
Tabela 5.18: *Prompts mais caros por modelo em ordem de custo, modelo Llama-3-70B-Instruct apresenta custos excepcionalmente superiores aos outros modelos. Os modelos adicionais estão destacados em cinza.*

Modelo	**Prompt**	**Custo por Inferência ($)**
Llama-3-70B-Instruct	zero-shot-none	0.025283
gpt-4-0125-preview	zero-shot-none	0.001382
Mistral-7B-Instruct-v0.3	cot-zero-shot-Toxic	0.001143
Gemma-7B-Instruct	few-shot-none	0.000633
gpt-4o	zero-shot-none	0.000434
Falcon-7B-Instruct	cot-few-shot-Not-Toxic	0.000414
Gemma-2B-Instruct	cot-zero-shot-both	0.000262
Llama-3-8B-Instruct	cot-zero-shot-none	0.000224
gpt-3.5-turbo-0125	zero-shot-none	0.000042

O modelo *Llama-3-8B-Instruct* mostrou os melhores resultados gerais, com o menor custo e maior acurácia. O modelo *GPT-4o*, que obteve resultados semelhantes ao melhor *prompt* do *Llama-3-8B-Instruct*, teve aproximadamente o dobro do custo, seguido do *Gemma-7b-Instruct*, exibiu custo cerca de três vezes maior nos *prompts* com desempenho similar ao primeiro colocado. O modelo *Mistral-7B-Instruct-v0.3* apresentou custo cerca de quatro vezes maior que o *Llama-3-8B-Instruct* em *prompts* com performance inferior.

Em uma análise apenas entre os melhores *prompts* por modelo, incluindo os modelos adicionais, podemos ver na figura 5.13 que os modelos *Llama-3-8B-Instruct* e *GPT-4o* apresentam a melhor relação custo por acurácia da dentre os modelos, seguido do modelo *Gemma-7b-Instruct*. Os modelo *GPT-3.5-turbo-0125* teve o menor custo e menor acurácia, seguido dos modelos *Gemma-2B-Instruct* e *Falcon-7B-Instruct*.

Figura 5.13: *Custo médio por inferência em Prompts com maior acurácia por modelo.*



Concluímos que os modelos *Llama-3-8B-Instruct*, *GPT-4o* e *Gemma-7B-Instruct* obtiveram resultados próximos e, numa análise estritamente de desempenho, seriam os candidatos a modelo final escolhido, nessa ordem, nos *prompts* definidos na figura 5.4. Contudo, a escolha de um modelo para o problema da pesquisa também precisa levar em conta a economicidade desses modelos e seu impacto no negócio.

Os modelos *Llama-3-8B-Instruct* no *prompt cot-zero-shot-none* e *GPT-4o* no *prompt zero-shot-none* apresentaram a melhor relação entre desempenho e custo dentre todos os modelos. Entretanto, há uma diferença significativa no custo médio por inferência entre eles, com o modelo *GPT-4o* custando cerca de 89% a mais, o que sugere a escolha do modelo *Llama-3-8B-Instruct*, conforme visto em tabela 5.19

Tabela 5.19: *Métricas de desempenho e custo médio por inferência em dólares americanos dos modelos candidatos finais*

Métrica	**Llama-3-8B-Instruct**	**GPT-4o**
Prompt	cot-zero-shot-none	zero-shot-none
Acurácia	**0,84**	0,83
Precisão	**0,84**	**0,84**
Recall	**0,84**	0,83
F1 Score	**0,84**	**0,84**
Custo Inferência ($)	**0,000224**	0,000434
Taxa de Alucinação	**0,005%**	0,011%

Na próxima seção, calcularemos e discutiremos o impacto no negócio da tarefa de classificação de toxicidade de ambos os LLMs, considerando o modelo matemático proposto na seção 4.1.10, visando a escolha final do modelo.

5.11 Análise de Impacto no Negócio

A proliferação de conversações tóxicas em jogos multijogador impacta significativamente na satisfação e retenção dos usuários, afetando, em última análise, a receita de uma empresa de jogos (KOWERT e KILMER, 2022; BERES *et al.*, 2021). A filtragem dessas conversas através da classificação incorre em custos e pode levar a classificação incorreta de conversações válidas, o que pode impactar a satisfação do usuário e, consequentemente, sua retenção.

Consideraremos a equação 4.19 do modelo proposto, conforme seção 4.1.10:

$$\Delta I = k_r k_s (k_t \alpha T - k_m M) - \bar{C}_T, \tag{5.9}$$

onde ΔI é a variação da receita dado a escolha do LLM, k_r é uma constante representando a receita gerada por usuário retido, k_s é a constante representando a taxa de retenção por unidade de satisfação do usuário, k_t é o impacto negativo por unidade de toxicidade, α são as probabilidades de classificação que impactam na toxicidade e T é o nível de toxicidade inicial. k_m é o impacto negativo por unidade de classificação incorreta, M a taxa de erros de classificação inserida pelo LLM e $\bar{C}_T$ é o custo médio de processamento da inferência.

Através da análise de desempenho, obtemos os valores α e M, e os valores de $\bar{C}_T$, durante a análise de custo. Assumiremos T, como $36,8\%$, de acordo com a distribuição de classes no conjunto de dados utilizados nesse trabalho, na seção 5.4. As variáveis k_r, k_s, k_t e k_m são variáveis relacionadas diretamente ao negócio, portanto, inacessíveis no contexto dessa pesquisa. Decidimos então, executar uma análise de sensibilidade do modelo para investigar o impacto destes parâmetros no ΔI. Op-

tamos por executar apenas a análise global de sensibilidade, pois buscamos uma avaliação mais abrangente da influência das variáveis de entrada no modelo, não apropriada para a análise local. É também mais apropriado para aplicações práticas, devido a variedade de condições nesses cenários (SOBOL, 2001).

5.11.1 Análise Global de Sensibilidade do Modelo

SOBOL (2001) propôs um método para análise de sensibilidade global baseado na execução de um modelo com uma amostra gerada sob certas premissas, e realiza uma decomposição de variância que permite o cálculo de um conjunto de índices de sensibilidade.

O índice de Sobol de primeira ordem avalia a contribuição individual de cada variável de entrada para a variância na saída do modelo, desconsiderando interações com outras variáveis. Em termos mais simples, ele quantifica a parcela da incerteza na saída do modelo que pode ser atribuída diretamente às mudanças em uma entrada específica, enquanto todas as outras entradas permanecem fixas. Essa avaliação é feita em uma escala de 0 a 1 (SOBOL, 2001).

O índice de Sobol de ordem total mede a contribuição geral de uma variável de entrada para a variância da saída, considerando tanto seu efeito individual, quanto suas interações com outras variáveis. Essencialmente, avalia o impacto de uma variável na saída, levando em conta todas as formas possíveis pelas quais ela pode influenciar o resultado, incluindo seus efeitos combinados com outras variáveis. Variando de 0 a 1, este índice indica a proporção da variabilidade da saída atribuída à influência total da variável de entrada (SOBOL, 2001).

O índice de Sobol de segunda ordem, e de ordens superiores, expande o índice de primeira ordem e é utilizado para medir o efeito combinado ou a interação entre pares, ou grupos, de variáveis de entrada na variância da saída do modelo. Enquanto o índice de primeira ordem se concentra na contribuição individual de cada variável, o índice de segunda ordem examina como a interação entre duas variáveis contribui para a incerteza na saída, além do que seria esperado com base nos efeitos individuais de cada variável (SOBOL, 2001). Isso proporciona uma representação visual clara do efeito combinado de pares de variáveis em uma matriz.

Portanto, uma análise de sensibilidade de Sobol (SOBOL, 2001) foi realizada utilizando o pacote SALib (IWANAGA *et al.*, 2022; HERMAN e USHER, 2017), em Python, usando os intervalos para as variáveis da equação 4.19, descritos na tabela 5.20. O racional utilizado para escolha do k_r foi o valor, em dólares americanos, que um jogador poderia gastar no jogo durante seu *Life Time Value* (LTV), os valores k_s, k_t e k_m representam o impacto da satisfação, toxicidade e por unidade de classificação incorreta, respectivamente, e foram definidos entre 0 e 1, indicando

nenhum impacto na métrica ou impacto total, respectivamente. Utilizamos 2^{20} amostras em cada execução.

Tabela 5.20: *Limites usados para cada variável na análise Sobol*

Variável	**Mínimo**	**Máximo**
k_r	0	1000
k_s	0	1
k_t	0	1
k_m	0	1

Executamos a análise de sensibilidade nos LLMs *Llama-3-8B-Instruct* e *GPT-4o* nos *prompts* selecionados, dado sua melhor relação entre custo por inferência por desempenho, conforme seção 5.10.2. As variáveis constantes são exibidas na tabela 5.21.

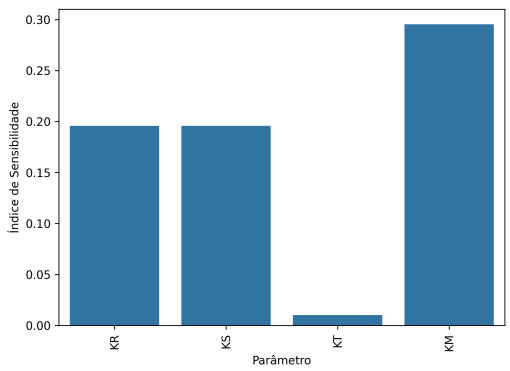
Tabela 5.21: *Variáveis constantes de cada modelo*

Variável	**Llama-3-8B-Instruct**	**GPT-4o**
α	0,3409	0,3405
M	0,6591	0,6595
$\bar{C}_T$	0,000224	0,000434

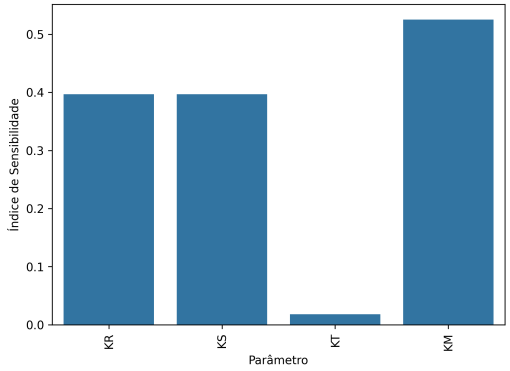
Podemos ver a análise de sensibilidade dos LLMs nas figuras 5.14 e 5.15. Dado a proximidade das constantes exibidas na tabela 5.21, os LLMs comportaram-se de maneira similar na análise de sensibilidade, em primeira, segunda e ordem total.

Da análise global de sensibilidade do modelo para ambos os LLMs, pode-se ver que k_m é o fator mais importante na análise de primeira ordem, seguido dos fatores k_r e k_s, com k_t tendo a menor impacto dentre os fatores, conforme nas figuras 5.14a e 5.15a. Nos índices de ordem total vemos o mesmo comportamento, com k_m, e o par k_r e k_s como fatores mais importantes, respectivamente. Essa relevância é vista nos índices de segunda ordem, conforme figuras 5.14c e 5.15c, onde é exibida a predominância dos pares k_r e k_s e sua relação com k_m.

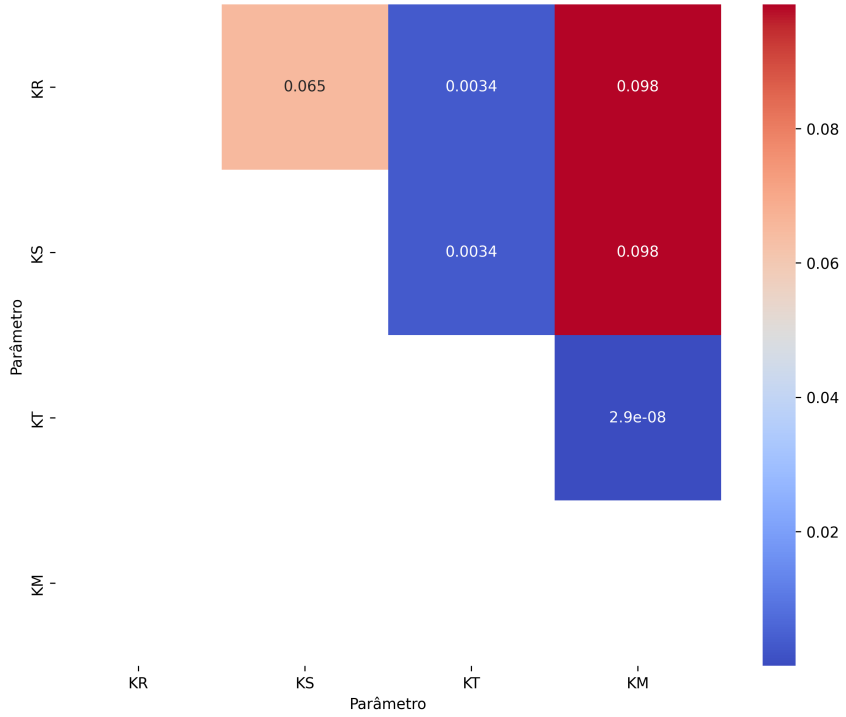
Figura 5.14: Análise de sensibilidade *Llama-3-8b-Instruct*



(a) Sensibilidade de Primeira Ordem

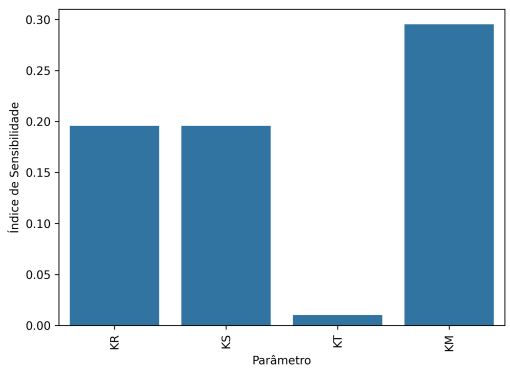


(b) Sensibilidade de Ordem Total

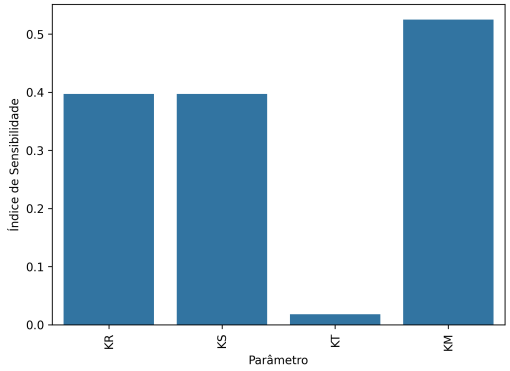


(c) Sensibilidade de Segunda Ordem

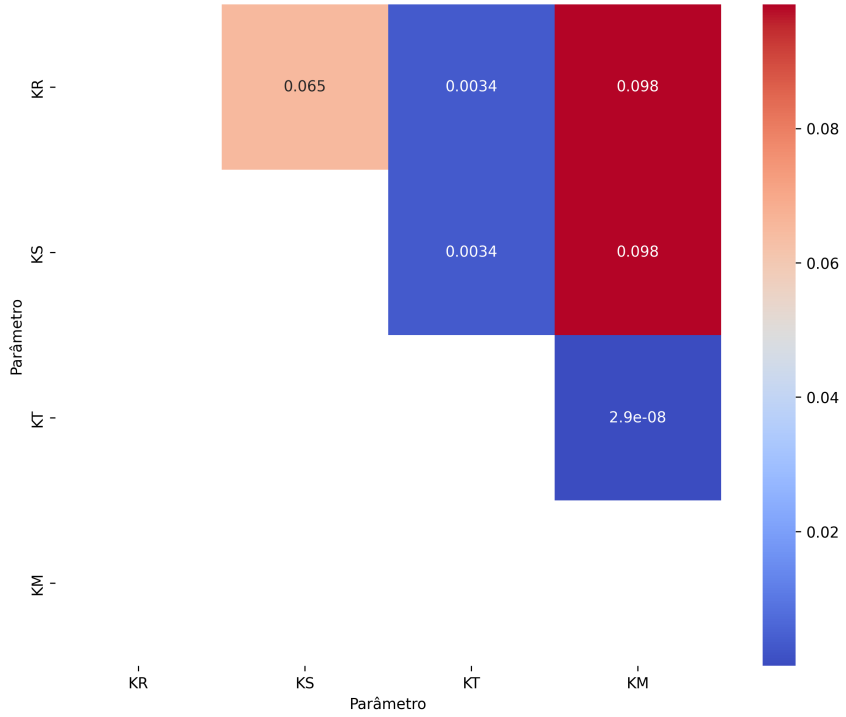
Figura 5.15: Análise de sensibilidade *GPT-4o*



(a) Sensibilidade de Primeira Ordem



(b) Sensibilidade de Ordem Total



(c) Sensibilidade de Segunda Ordem

Dentro dos limites utilizados, a análise de sensibilidade do modelo mostra que a variação de renda devido a escolha de um LLM (ΔI) é particularmente sensível a k_m, o impacto negativo por unidade de classificação incorreta do modelo. Em termos de negócio, isso demonstra uma atenção particular a métricas de desempenho na escolha do *LLM*, dado que M é uma constante medida através de *benchmarks* na tarefa, tornando o desempenho do LLM escolhido como o principal fator na hora da escolha final.

O par k_r e k_s se mostrou como o segundo fator em relevância na escolha, e representa uma relação entre a retenção do usuário dada sua satisfação no jogo e a receita gerada por usuário retido. Em termos de negócio, podemos associar essas constantes com o *Life Time Value* (LTV) esperado de um jogador, demonstrando

que a satisfação do jogador e sua retenção tem um grande impacto no ΔI, racional amplamente discutido no decorrer deste trabalho.

Na próxima seção, discutiremos a escolha final do LLM a ser usado, considerando os resultados obtidos nas seções 5.9 e 5.10.

5.12 Escolha de LLM

Considerando os resultados de desempenho, custo e a análise de impacto no negócio dentre os LLMs finais, selecionamos o modelo *Llama-3-8B-Instruct*. Considerando as semelhanças, o fator preponderante da decisão foi o custo de inferência. Ele apresentou a melhor relação entre desempenho e custo de inferência, com custo e taxas de alucinação aproximadamente 50% menores em relação ao *GPT-4o*, conforme pode ser visto na tabela 5.22.

Tabela 5.22: Comparação de Métricas entre Llama-3-8B-Instruct e GPT-4o

Métrica	**Llama-3-8B-Instruct**	**GPT-4o**	**Diferença Relativa (%)**
Parâmetros	8 bilhões	**1,8 trilhão**	22500%
F1 Score	**0,84**	**0,84**	0,00
Custo Inferência ($)	**0,000224**	0,000434	48,39
Taxa de Alucinação	**0,005%**	0,011%	54,55

Essa escolha destaca a relevância dos SLM, dado a diferença massiva na diferença nos parâmetros dos modelos, sendo o *GPT-4o* cerca de 225 vezes maior que o *Llama-3-8B-Instruct*. Os resultados encontrados indicam avanços significativos no treinamento e desenvolvimento dos SLM, demonstrando resultados comparáveis, ou até mesmo superiores a modelos com um número de parâmetros muito superiores. Demonstra também que os retornos decrescentes no número de parâmetros dos LLM são significativos para algumas tarefas através de *transfer learning*.

Encerrada a etapa exploratória e subsequente escolha do LLM, a próxima etapa é o seu potencial uso em um ambiente de produção. Para entender melhor as limitações deste LLM, analisaremos os resultados do *prompt* escolhido *zero-shot-cot-none* utilizando a base em sua completude. Esta última etapa visa alinhar as expectativas do negócio quanto ao uso do LLM, e busca documentar de forma mais abrangente os cenários onde o LLM funciona adequadamente ou não, conforme definido na seção 5.9.

5.12.1 Análise dos Resultados

A tabela 5.23 mostra as métricas de desempenho do modelo quando executado na base completa. Podemos ver que o modelo comporta-se de forma semelhante

ao esperado numa amostragem maior, tendo um desempenho semelhante a etapa exploratória, corroborando seu uso em ambiente em um ambiente mais extensivo como o de produção.

Tabela 5.23: *Métricas de desempenho Llama-3-8B-Instruct*

Modelo	**F1**	**Precisão**	**Recall**	**Acurácia**
Llama-3-8B-Instruct	0,85	0,85	0,86	0,85

Para uma análise mais detalhada, examinaremos alguns exemplos específicos em que o classificador funcionou bem e outros onde ele falhou. Esta análise permitirá identificar os pontos fortes do modelo, bem como as áreas que necessitam de melhorias. Esses exemplos servirão como referência para trabalhos futuros e para ajustes no uso do LLM em produção.

A base de dados usada, CONDA (WELD *et al.*, 2021), discutida na seção 5.4, foi rotulada através de *crowdsourcing*, e durante a análise foi possível notar alguns erros nos rótulos, conforme pode ser visto na tabela 5.24. Considerando ser uma base na língua inglesa, foram encontradas também algumas instâncias de outras linguagens, como português ou espanhol, *e.g.* "como essa templar caga mds", "ATRAS RATAS ATRAS".

Tabela 5.24: Exemplos de conversações da base de dados com erros

Conversação	**Base**	**Tóxico**
damn son	✓	✗
Ez mid	✓	✗
holy shit gggggggggg	✓	✗

O classificador demonstrou um bom desempenho em conversações onde a toxicidade da conversação era clara, com uso de palavras consideradas tóxicas:

```
- f9  faggot viper  fuck your mom he coming [...]
- safe to leave really noob? shut up noob fuck you [...]
- FU ALCHE
- going afk this shit team im realy considering
```

Contudo, o classificador parece ter uma sensibilidade muito alta a palavras isoladas e não utiliza o contexto da conversação, classificando-as como tóxica diretamente. Críticas ou reclamações pessoais, como por exemplo solicitar um *report*, autodepreciação em expressões como *"Fuck me"* ou *"Fuck this"*, ou até pequenas demonstrações de toxicidade, como chamar alguém de *noob* também estão sendo incorretamente classificadas como tóxicas:

```
- Fuck me
- NO FUCK YOUR WOLF
- so noob this trx ggwp gg
- pls report yellow for ruining game on purpose  thanks
```

Dado a análise das respostas do classificador, fica claro que ele exibe desempenho apropriado em conversações com a toxicidade bem definida, porém, não utiliza o contexto da conversação adequadamente, focando em palavras onde o LLM pré-treinado conhece a toxicidade. Na análise de sensibilidade, vimos que a classificação incorreta de toxicidade tem o maior impacto na variação de receita do modelo, indicando que em casos onde a toxicidade é clara, o uso do LLM é apropriado, porém em casos mais sutis o uso de moderação humana pode ser adequado.

Finalmente, concluímos que o modelo *Llama-3-8B-Instruct* demonstrou métricas de desempenho adequadas, superando marginalmente os resultados encontrados por YANG *et al.* (2023), que comparou a mesma base de dados deste trabalho com outras APIs estabelecidas, como *CleanSpeak*[12], *Detoxify*[13] e *PerspectiveAPI*[14]. Os resultados indicam também que um LLM generalista aprendem tarefas de uso final por *transfer learning*, com desempenho satisfatório, conforme proposto por BROWN *et al.* (2020). O uso de técnicas de engenharia de *prompt* pode obter resultados satisfatórios em tarefas de uso final, embora a multitude de técnicas disponíveis tornem a seleção de um *prompt* adequado em uma tarefa complexa e custosa, afetando grandemente no desempenho da LLM na tarefa.

[12] https://cleanspeak.com/docs/3.x/tech/apis/
[13] https://github.com/unitaryai/detoxify
[14] https://perspectiveapi.com

Capítulo 6

Conclusões

Neste trabalho, definimos uma metodologia para escolha de LLMs para tarefas de uso final e propomos um modelo de impacto no negócio da escolha de um LLM na tarefa de classificação de toxicidade. Dado o uso de LLMs, selecionamos técnicas de Engenharia de *Prompt* para elaborar um total de dezesseis *prompts* para análise cada LLM, num total de dez LLMs.

Durante a escolha do LLM a ser usado, efetuamos uma análise de desempenho, custo e impacto da escolha no negócio. Na análise de custo, relacionamos o custo por inferência com os resultados de desempenho de todos os modelos, selecionando os modelos *Llama-3-8B-Instruct* e *GPT-4o*. Finalmente, analisamos o impacto no negócio da escolha de ambos os modelos.

Para a análise de impacto do negócio, propomos um modelo que relaciona principalmente a satisfação do usuário como pilar na retenção, impactando o *Life Time Value* (LTV) daquele jogador e consequentemente a receita da desenvolvedora. Através da análise de sensibilidade desse modelo, aprendemos que a variação de receita é particularmente sensível a classificações incorretas do modelo, destacando a importância das métricas de performance na escolha. Por fim, selecionamos o LLM *Llama-3-8B-Instruct*, provando-se adequado a tarefa.

6.1 Resultados e contribuições

A metodologia proposta mostrou-se apropriada para uso na seleção de LLMs para a tarefa de classificação, com etapas estruturadas desde a elaboração do cenário do negócio, até a definição do impacto da sua escolha no mesmo. A análise de desempenho demonstrou o impacto dos diferentes *prompts* na classificação de toxicidade, e mostrou também a complexidade na elaboração de um *prompt* adequado, dado o número de técnicas disponíveis na emergente Engenharia de *Prompt*.

Através da analise de sensibilidade do modelo proposto, vimos que a classificação incorreta de toxicidade tem o maior impacto na variação de receita do modelo, se-

guido da satisfação do usuário e retenção do mesmo, racional amplamente discutido neste trabalho.

Nossa avaliação experimental demonstrou que LLMs generalistas aprendem a tarefa de classificação de toxicidade por *transfer learning* com desempenho satisfatório, especialmente em conversações onde a toxicidade é clara, entretanto, possui sensibilidade muito altas a palavras isoladas e não a conversação como um todo.

Logo, resumimos algumas contribuições deste trabalho:

(i) Uma metodologia estruturada para a escolha e implementação de LLMs em tarefas de uso final, como classificação

(ii) Avaliação de desempenho de *prompts* com diferentes estratégias, *e.g Chain-of-Thought, Few-Shot* e variações quanto definição dos rótulos em dez diferentes LLMs

(iii) Uma discussão quanto às alucinações de LLMs em diferentes *prompts* e LLMs

(iv) Avaliação do custo de diferentes LLMs e SLMs

(v) Um modelo matemático de impacto no negócio causado pela escolha do LLM na tarefa de classificação de toxicidade

(vi) Discussão sobre as limitações e casos de borda do LLM final escolhido

6.2 Trabalhos Futuros

Durante a elaboração deste trabalho, diversas oportunidades para pesquisas futuras foram identificadas. Algumas dessas oportunidades são listadas a seguir:

(i) Exploração de Modelos Não Lineares: O modelo de impacto no negócio proposto baseia-se em relações lineares entre variáveis. Futuras pesquisas podem investigar modelos não lineares para capturar melhor as complexidades e dinâmicas envolvidas na classificação de toxicidade e seu impacto nos negócios.

(ii) Exploração de Novos Modelos: Com o rápido avanço da tecnologia de LLMs, novos modelos estão continuamente sendo desenvolvidos. Isso inclui a avaliação de novos LLMs e SLMs em termos de desempenho, custo e impacto no negócio.

(iii) Uso de *Fine-Tuning* nos Modelos: Embora a pesquisa atual tenha utilizado técnicas de engenharia de *prompt*, dada as restrições de tempo e custo, trabalhos futuros podem explorar o *fine-tuning* dos modelos.

(iv) Integração com Outras Ferramentas de Moderação: A combinação de LLMs com outras ferramentas de moderação de conteúdo pode ser investigada. Isso inclui a integração com sistemas baseados em regras e a utilização de moderação humana para casos de borda, criando um sistema de moderação híbrido que maximiza a eficácia e minimiza os erros.

(v) Geração de *Prompts*: Dado o grande número de estratégias de *prompt* disponíveis, o uso de abordagens como o *Automatic Prompt Engineer* (APE) (ZHOU *et al.*, 2023) e *Prompt Space* (SHI *et al.*, 2024), que utilizam de formas automatizadas de derivar o *prompt*, pode ser explorado em trabalhos futuros

(vi) Exploração com LLMs híbridos: Dado a relação desempenho e custo dos LLMs, o uso de diferentes LLMs dinamicamente durante a inferência pode representar um equilíbrio entre custo e qualidade da resposta, conforme proposto por ONG *et al.* (2024)

Referências Bibliográficas

ADL. "Hate Is No Game Hate and Harassment in Online Games 2022 The state of hate, harassment, and extremism in online multiplayer games". 2023. Disponível em: <https://www.adl.org/resources/report/hate-no-game-hate-and-harassment-online-games-2022>.

OECD. "AI language models", *OECD digital economy papers*, 2023. doi: https://doi.org/https://doi.org/10.1787/13d38f92-en. Disponível em: <https://www.oecd-ilibrary.org/content/paper/13d38f92-en>.

BROWN, T. B., MANN, B., RYDER, N., et al. "Language Models are Few-Shot Learners". 2020. Disponível em: <https://arxiv.org/abs/2005.14165>.

PRINCE, S. J. D. "Understanding Deep Learning". In: *Understanding Deep Learning*, cap. 12, pp. 207–216, USA, The MIT Press, 2024. Disponível em: <libgen.li/file.php?md5=7e712994461c43ef382fa07463e39238>.

SAHOO, P., SINGH, A. K., SAHA, S., et al. "A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications". 2024. Disponível em: <https://arxiv.org/abs/2402.07927>.

KOWERT, R., KILMER, E. "TOXIC GAMERS ARE ALIENATING YOUR CORE DEMOGRAPHIC: THE BUSINESS CASE FOR COMMUNITY MANAGEMENT". 2022.

GDC. "GDC 2024 State of the Game Industry Report". 2024. Disponível em: <https://reg.gdconf.com/state-of-game-industry-2024>. (Accessed on 06/03/2024).

SCHMIDT, A., WIEGAND, M. "A Survey on Hate Speech Detection using Natural Language Processing". 2017. Disponível em: <https://en.wikipedia.org/wiki/List_>.

VASWANI, A., SHAZEER, N., PARMAR, N., et al. "Attention Is All You Need". 2023. Disponível em: <https://arxiv.org/abs/1706.03762>.

NAVEED, H., KHAN, A. U., QIU, S., et al. “A Comprehensive Overview of Large Language Models”. 2024. Disponível em: <https://arxiv.org/abs/2307.06435>.

LEPAGNOL, P., GERALD, T., GHANNAY, S., et al. “Small Language Models are Good Too: An Empirical Study of Zero-Shot Classification”. 2024. Disponível em: <https://arxiv.org/abs/2404.11122>.

TOUVRON, H., LAVRIL, T., IZACARD, G., et al. “LLaMA: Open and Efficient Foundation Language Models”. 2023. Disponível em: <https://arxiv.org/abs/2302.13971>.

FROMMEL, J., MANDRYK, R. “Effective Toxicity Prediction in Online Multiplayer Gaming: Four Obstacles to Making Approaches Usable”. Mensch und Computer 2022 - Workshopband, 2022.

FOUNTA, A.-M., DJOUVAS, C., CHATZAKOU, D., et al. “Large Scale Crowdsourcing and Characterization of Twitter Abusive Behavior”. 2018. Disponível em: <https://arxiv.org/abs/1802.00393>.

HE, X., ZANNETTOU, S., SHEN, Y., et al. “You Only Prompt Once: On the Capabilities of Prompt Learning on Large Language Models to Tackle Toxic Content”. 2023. Disponível em: <https://arxiv.org/abs/2308.05596>.

WELD, H., HUANG, G., LEE, J., et al. “CONDA: a CONtextual Dual-Annotated dataset for in-game toxicity understanding and detection”. 2021. Disponível em: <https://arxiv.org/abs/2106.06213>.

SULER, J. “The online disinhibition effect”. 6 2004. ISSN: 10949313.

HEMENDINGER, E. “The harmful effects of social media use on mental health, including body image and development of eating disorders”. 6 2023. Disponível em: <https://medicalxpress.com/news/2023-06-effects-social-media-mental-health.html>.

KOWERT, R., OLDMEADOW, J. A. “Playing for social comfort: Online video game play as a social accommodator for the insecurely attached”, *Computers in Human Behavior*, v. 53, pp. 556–566, 2015. ISSN: 0747-5632. doi: https://doi.org/10.1016/j.chb.2014.05.004. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0747563214002829>.

BERES, N. A., FROMMEL, J., REID, E., et al. “Don’t you know that you’re toxic: Normalization of toxicity in online gaming”, *Conference on Human*

Factors in Computing Systems - Proceedings, 5 2021. doi: 10.1145/3411764.3445157.

ZHANG, J., CHANG, J. P., DANESCU-NICULESCU-MIZIL, C., et al. "Conversations Gone Awry: Detecting Early Signs of Conversational Failure". 2018. Disponível em: <http://convokit.infosci.cornell.edu>.

BLACKBURN, J., KWAK, H. "STFU NOOB! Predicting crowdsourced decisions on toxic behavior in online games", *WWW 2014 - Proceedings of the 23rd International Conference on World Wide Web*, pp. 877–887, 4 2014. doi: 10.1145/2566486.2567987.

HOWE, JEFF. "The Rise of Crowdsourcing", *Wired*, v. 14, 01 2006.

KAZBEKOVA, G., ISMAGULOVA, Z., KEMELBEKOVA, Z., et al. "Offensive Language Detection on Online Social Networks using Hybrid Deep Learning Architecture", *International Journal of Advanced Computer Science and Applications*, v. 14, 01 2023. doi: 10.14569/IJACSA.2023.0141180.

AGRAWAL, S., AWEKAR, A. "Deep Learning for Detecting Cyberbullying Across Multiple Social Media Platforms". 2018. Disponível em: <https://arxiv.org/abs/1801.06482>.

DEVLIN, J., CHANG, M.-W., LEE, K., et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". 2019. Disponível em: <https://arxiv.org/abs/1810.04805>.

MARTENS, M., SHEN, S., IOSUP, A., et al. "Toxicity detection in multiplayer online games". 1 2016. ISSN: 21568146.

DE MESQUITA NETO, J. A., BECKER, K. "Relating conversational topics and toxic behavior effects in a MOBA game", *Entertainment Computing*, v. 26, pp. 10–29, 2018. ISSN: 1875-9521. doi: https://doi.org/10.1016/j.entcom.2017.12.004. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1875952117300824>.

CHEN, Q., ZHUO, Z., WANG, W. "BERT for Joint Intent Classification and Slot Filling". 2019. Disponível em: <https://arxiv.org/abs/1902.10909>.

YANG, Z., MARICAR, Y., DAVARI, M., et al. "ToxBuster: In-game Chat Toxicity Buster with BERT". 2023. Disponível em: <https://arxiv.org/abs/2305.12542>.

KOH, H., KIM, D., LEE, M., et al. “Can LLMs Recognize Toxicity? Definition-Based Toxicity Metric”. 2024. Disponível em: <https://arxiv.org/abs/2402.06900>.

DE WYNTER, A., WATTS, I., ALTINTOPRAK, N. E., et al. “RTP-LX: Can LLMs Evaluate Toxicity in Multilingual Scenarios?” 2024. Disponível em: <https://arxiv.org/abs/2404.14397>.

RADFORD, A., WU, J., CHILD, R., et al. “Language Models are Unsupervised Multitask Learners”. 2019. Disponível em: <https://api.semanticscholar.org/CorpusID:160025533>.

RAFFEL, C., SHAZEER, N., ROBERTS, A., et al. “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer”. 2023. Disponível em: <https://arxiv.org/abs/1910.10683>.

YANG, Z., DAI, Z., YANG, Y., et al. “XLNet: Generalized Autoregressive Pretraining for Language Understanding”. 2020. Disponível em: <https://arxiv.org/abs/1906.08237>.

MIKOLOV, T., CHEN, K., CORRADO, G., et al. “Efficient Estimation of Word Representations in Vector Space”. 2013. Disponível em: <https://arxiv.org/abs/1301.3781>.

LIU, P., YUAN, W., FU, J., et al. “Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing”, *ACM Computing Surveys*, v. 55, 1 2023. ISSN: 15577341. doi: 10.1145/3560815.

TONMOY, S. M. T. I., ZAMAN, S. M. M., JAIN, V., et al. “A Comprehensive Survey of Hallucination Mitigation Techniques in Large Language Models”. 2024. Disponível em: <https://arxiv.org/abs/2401.01313>.

CHEN, B., ZHANG, Z., LANGRENÉ, N., et al. “Unleashing the potential of prompt engineering in Large Language Models: a comprehensive review”. 2024. Disponível em: <https://arxiv.org/abs/2310.14735>.

AMATRIAIN, X. “Prompt Design and Engineering: Introduction and Advanced Methods”. 2024. Disponível em: <https://arxiv.org/abs/2401.14423>.

WEI, J., WANG, X., SCHUURMANS, D., et al. “Chain-of-Thought Prompting Elicits Reasoning in Large Language Models”. 2023. Disponível em: <https://arxiv.org/abs/2201.11903>.

LI, X. L., LIANG, P. “Prefix-Tuning: Optimizing Continuous Prompts for Generation”. 2021. Disponível em: <https://arxiv.org/abs/2101.00190>.

TANG, T., LI, J., ZHAO, W. X., et al. “Context-Tuning: Learning Contextualized Prompts for Natural Language Generation”. 2022. Disponível em: <https://arxiv.org/abs/2201.08670>.

KOJIMA, T., GU, S. S., REID, M., et al. “Large Language Models are Zero-Shot Reasoners”. 2023. Disponível em: <https://arxiv.org/abs/2205.11916>.

MIN, S., LYU, X., HOLTZMAN, A., et al. “Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?” 2022. Disponível em: <https://arxiv.org/abs/2202.12837>.

VON ROSING, M., WHITE, S., CUMMINS, F. A., et al. “Business Process Model and Notation - BPMN”. In: *The Complete Business Process Handbook, Vol. I*, 2015. Disponível em: <https://api.semanticscholar.org/CorpusID:36184220>.

RUBLE, D. A. “Practical analysis and design for client/server and GUI systems”. In: *Practical analysis and design for client/server and GUI systems*, cap. 2, pp. 36–40, USA, Prentice-Hall, Inc., 1997. ISBN: 013521758X.

GANE, C. P., SARSON, T. “Structured Systems Analysis: Tools and Techniques”. In: *Structured systems analysis : tools and techniques*, 1st ed., cap. 1, pp. 5–9, USA, Prentice Hall Professional Technical Reference, 1979. ISBN: 0138545472.

DING, B., QIN, C., ZHAO, R., et al. “Data Augmentation using Large Language Models: Data Perspectives, Learning Paradigms and Challenges”. 2024. Disponível em: <https://arxiv.org/abs/2403.02990>.

SHI, F., QING, P., YANG, D., et al. “Prompt Space Optimizing Few-shot Reasoning Success with Large Language Models”. 2024. Disponível em: <https://arxiv.org/abs/2306.03799>.

ZHOU, Y., MURESANU, A. I., HAN, Z., et al. “Large Language Models Are Human-Level Prompt Engineers”. 2023. Disponível em: <https://arxiv.org/abs/2211.01910>.

OUYANG, L., WU, J., JIANG, X., et al. “Training language models to follow instructions with human feedback”, *Advances in neural information processing systems*, v. 35, pp. 27730–27744, 2022.

HUANG, Y., XU, J., LAI, J., et al. “Advancing Transformer Architecture in Long-Context Large Language Models: A Comprehensive Survey”. 2024. Disponível em: <https://arxiv.org/abs/2311.12351>.

TEAM, O. “OpenDota - Dota 2 Statistics”. 2024. Disponível em: <https://www.opendota.com/>. [Online; acessado 23-January-2024].

GOOGLE, D. “Gemma: Google introduces new state-of-the-art open models”. 02 2024. Disponível em: <https://blog.google/technology/developers/gemma-open-models/>.

DONG, Y., MU, R., ZHANG, Y., et al. “Safeguarding Large Language Models: A Survey”. 2024. Disponível em: <https://arxiv.org/abs/2406.02622>.

SOBOL, I. M. “Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates”, *Mathematics and computers in simulation*, v. 55, n. 1-3, pp. 271–280, 2001.

IWANAGA, T., USHER, W., HERMAN, J. “Toward SALib 2.0: Advancing the accessibility and interpretability of global sensitivity analyses”, *Socio-Environmental Systems Modelling*, v. 4, pp. 18155, 5 2022. doi: 10.18174/sesmo.18155. Disponível em: <https://sesmo.org/article/view/18155>.

HERMAN, J., USHER, W. “SALib: An open-source Python library for Sensitivity Analysis”, *The Journal of Open Source Software*, v. 2, n. 9, 1 2017. doi: 10.21105/joss.00097. Disponível em: <https://doi.org/10.21105/joss.00097>.

ONG, I., ALMAHAIRI, A., WU, V., et al. “RouteLLM: Learning to Route LLMs with Preference Data”. 2024. Disponível em: <https://arxiv.org/abs/2406.18665>.

Apêndice A

Diagrama de Processo Expandido

A figura A.1 representa o diagrama BPMN completo e expandido da metodologia proposta no capítulo 4. As caixas em destaque possuíram maior enfoque neste trabalho, dada sua natureza.

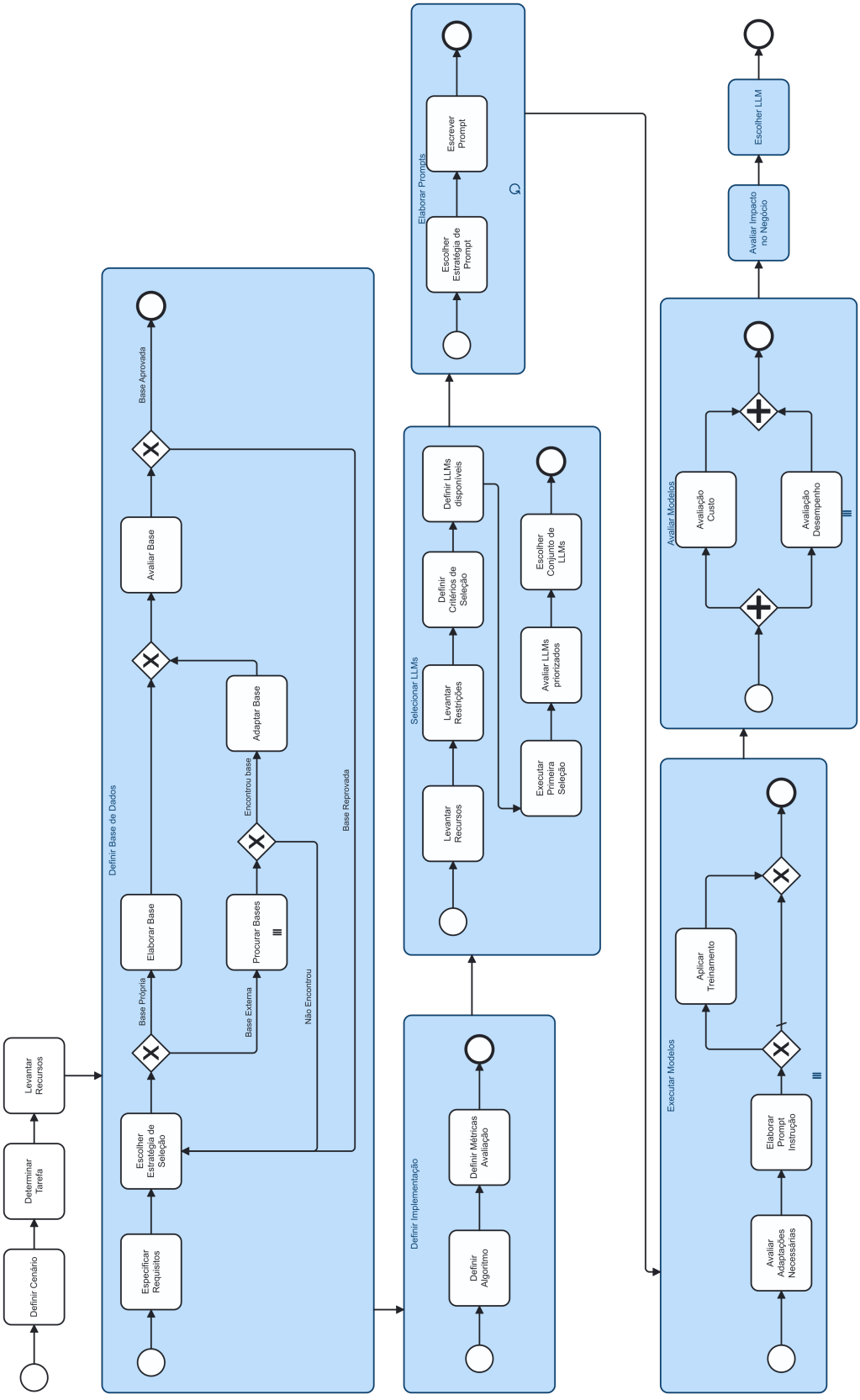


Figura A.1: Diagrama BPMN de processo expandido

Apêndice B

Prompts usados

As listagens B.1, B.2, B.3 e B.4 contém os prompts utilizados nos experimentos.

Listagem B.1: Prompt Zero-Shot

```
Classify the following Dota 2 chat message as 'Toxic' or
'Not-Toxic':
Message: {{message}}.
```

Listagem B.2: Prompt Few-Shot

```
Classify the following Dota 2 chat messages as 'Toxic' or
'Not-Toxic':
Example 1: Message: 'Good job team, we can win this!' Conclusion:
Not-Toxic Example 2: Message: 'You are completely useless, go back
to noob games.' Conclusion: Toxic Example 3: Message: 'Nice play!
Let's keep up the pressure.' Conclusion: Not-Toxic Example 4:
Message: 'Why are you even playing? You're garbage.' Conclusion:
Toxic
Now classify the following message:
Message: {{message}} Conclusion:
```

Listagem B.3: Prompt Zero-Shot com Chain-of-Thought

```
Definition of Toxic: A message is considered Toxic if it contains
personal attacks, insults, offensive language, or hostile and
negative tones.
Definition of Not-Toxic: A message is considered Not-Toxic if it
does not contain personal attacks, insults, offensive language, or
hostile and negative tones. Instead, it may be neutral, positive,
or encouraging.
```

Listagem B.4: Prompt Few-Shot com Chain-of-Thought

```
Classify the following Dota 2 chat message as 'Toxic' or
'Not-Toxic' by reasoning through the content step by step:n
Message 1: "You are the worst player I've ever seen, uninstall the
```

```
game." Let's think step by step: 1. The message contains a
personal attack by calling the player the worst and suggesting
they uninstall the game. 2. The language used ("worst player,"
"uninstall the game") is offensive and derogatory. 3. The overall
tone of the message is hostile and negative. Conclusion: Toxic ---
Message 2: "Nice job on that last play, keep it up!" Let's think
step by step: 1. The message contains positive reinforcement by
complimenting the player's performance. 2. The language used
("Nice job," "keep it up") is encouraging and supportive. 3. The
overall tone of the message is positive and friendly. Conclusion:
Not-Toxic --- Message 3: "You need to learn how to play properly,
you're making us lose." Let's think step by step: 1. The message
contains criticism of the player's skill level. 2. The language
used ("learn how to play properly," "making us lose") is critical
but not overtly offensive. 3. The overall tone of the message is
negative and somewhat hostile. Conclusion: Toxic --- Message 4:
"Good effort, but we need to work on our coordination." Let's
think step by step: 1. The message contains constructive feedback
about team coordination. 2. The language used ("Good effort," "we
need to work on") is positive and focused on improvement. 3. The
overall tone of the message is constructive and supportive.
Conclusion: Not-Toxic --- Message: {{message}}. Return only the
Classification for the last message:
```

As listagens B.5, B.6 e B.7 contém as definições de toxicidade, às quais os prompts são concatenados em alguns experimentos.

Listagem B.5: Definições de toxicidade

```
Definition of Toxic: A message is considered Toxic if it contains
personal attacks, insults, offensive language, or hostile and
negative tones.
```

Listagem B.6: Definições de não-toxicidade

```
Definition of Not-Toxic: A message is considered Not-Toxic if it
does not contain personal attacks, insults, offensive language, or
hostile and negative tones. Instead, it may be neutral, positive,
or encouraging.
```

Listagem B.7: Definições de toxicidade e não-toxicidade

```
Definition of Toxic: A message is considered Toxic if it contains
personal attacks, insults, offensive language, or hostile and
negative tones.\nDefinition of Not-Toxic: A message is considered
Not-Toxic if it does not contain personal attacks, insults,
offensive language, or hostile and negative tones. Instead, it may
be neutral, positive, or encouraging.
```

Apêndice C

Códigos usados

A listagem C.1 é o código utilizado para avaliação dos modelos disponíveis através da plataforma Hugging Face.

Listagem C.1: Código responsável pela avaliação dos modelos provenientes da plataforma Hugging Face e cálculos das métricas de análise.

```
from transformers import (
    AutoModelForSequenceClassification,
    Trainer,
    TrainingArguments,
    AutoModelForCausalLM,
)
import transformers
from transformers import GPT2Tokenizer
from datasets import load_from_disk
import csv
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score
from sagemaker.predictor import Predictor, retrieve_default
import json
from tqdm import tqdm
import time
from utils import (
    generate_instruction_prompt,
    prepare_prompt_list,
    process_csv,
    get_generated_text,
)

models = {
    "gemma-7b": {
        "model_id": "google/gemma-7b",
        "model_name": "Gemma-7B-Instruct",
        "model_family": "gemma",
        "endpoint_name": "jumpstart-dft-hf-llm-gemma-7b-instr-20240602-190219",
        "new": False,
    },
    "gemma-2b": {
        "model_id": "google/gemma-2b",
        "model_name": "Gemma-2B-Instruct",
        "model_family": "gemma",
        "endpoint_name": "jumpstart-dft-hf-llm-gemma-2b-instr-20240602-180028",
        "new": False,
```

```
    },
    "mistral-7b": {
        "model_id": "mistralai/Mistral-7B-Instruct-v0.3",
        "model_name": "Mistral-7B-Instruct-v0.3",
        "model_family": "mistral",
        "endpoint_name": "jumpstart-dft-hf-llm-mistral-7b-ins-20240603-180820",
        "inference_component_name":
"huggingface-llm-mistral-7b-instruct-20240603-180822",
        "new": True,
    },
    "mistral-8x7b": {
        "model_id": "mistralai/Mixtral-8x7B-Instruct-v0.1",
        "model_name": "Mistral-8x7B-Instruct-v0.1",
        "model_family": "mistral",
        "endpoint_name": "jumpstart-dft-hf-llm-mixtral-8x7b-i-20240606-192056",
        "new": False,
    },
    "llama-3-8b": {
        "model_id": "meta-llama/Meta-Llama-3-8B-Instruct",
        "model_name": "Llama-3-8B-Instruct",
        "model_family": "llama3",
        "endpoint_name": "jumpstart-dft-meta-textgeneration-l-20240603-220737",
        "inference_component_name":
"meta-textgeneration-llama-3-8b-instruct-20240603-220738",
        "new": True,
    },
    "llama-3-70b": {
        "model_id": "meta-llama/Meta-Llama-3-70B-Instruct",
        "model_name": "Llama-3-70B-Instruct",
        "model_family": "llama3",
        "endpoint_name": "jumpstart-dft-meta-textgeneration-l-20240606-134651",
        "inference_component_name":
"meta-textgeneration-llama-3-70b-instruct-20240606-134652",
        "new": True,
    },
    "falcon-7b-instruct": {
        "model_id": "tiiuae/falcon-7b-instruct",
        "model_name": "Falcon-7B-Instruct",
        "model_family": "falcon",
        "endpoint_name": "jumpstart-dft-hf-llm-falcon-7b-inst-20240605-135024",
        "inference_component_name":
"huggingface-llm-falcon-7b-instruct-bf16-20240605-135027",
        "new": True,
    },
    "falcon-40b-instruct": {
        "model_id": "tiiuae/falcon-40b-instruct",
        "model_name": "Falcon-40B-Instruct",
        "model_family": "falcon",
        "endpoint_name": "jumpstart-dft-hf-llm-falcon-40b-ins-20240606-135155",
        "inference_component_name":
"huggingface-llm-falcon-40b-instruct-bf16-20240606-135156",
        "new": True,
    },
}

current_model = models["mistral-8x7b"]

tokenizer = GPT2Tokenizer.from_pretrained("gpt2")
# Load your test dataset
```

```
# test_dataset = load_from_disk("../dataset/test")
test_dataset = load_from_disk("../dataset/test").select(range(3))
model_id = current_model.get("model_id")
model_name = current_model.get("model_name")
model_family = current_model.get("model_family")

endpoint_name = current_model.get("endpoint_name")

parameters = generate_instruction_prompt("", model_family, "test")["parameters"]

prompt_list = prepare_prompt_list()

predictor = (
    Predictor(endpoint_name)
    if current_model.get("new") == False
    else retrieve_default(
        endpoint_name=endpoint_name,
        inference_component_name=current_model.get("inference_component_name"),
    )
)

predictor.content_type = "application/json"
predictor.accept = "application/json"

current_time = time.strftime("%Y-%m-%dT%H:%M:%S")

for key, prompts in prompt_list.items():
    for prompt in prompts:
        print(
            f"Running predictions for {model_name} with prompt {key} and
definitions: {prompt['definition']}\n\n"
        )

        prompts = [
            generate_instruction_prompt(
                prompt["prompt"], model_family, sample["context"]
            )["prompt"]
            for sample in test_dataset
        ]

        context = [str(sample["context"]) for sample in test_dataset]
        references = [str(sample["completion"]) for sample in test_dataset]

        definition = prompt["definition"]
        filename = f"predictions-{model_name}-{key}-{definition}-{current_time}"

        with open(
            f"../results/raw/{model_name}/{filename}.csv", "w", newline=""
        ) as file:
            writer = csv.writer(file)
            writer.writerow(
                ["reference", "prediction", "input_tokens", "output_tokens"]
            )

            predictions = []

            # count the time it's taking to run this loop
            startTime = time.time()
            for prompt, reference, context in tqdm(
```

```
                zip(prompts, references, context), total=len(prompts)
            ):
                payload = {"inputs": prompt, "parameters": parameters}

                final_payload = (
                    payload if current_model.get("new") == True else
json.dumps(payload)
                )

                res = predictor.predict(final_payload)

                generated_text = get_generated_text(
                    res, current_model.get("new"),
current_model.get("model_family")
                )

                input_tokens = len(tokenizer.encode(prompt))
                output_tokens = len(tokenizer.encode(generated_text))

                writer.writerow(
                    [reference, generated_text, input_tokens, output_tokens]
                )

                predictions.append(generated_text)

            with open(
                f"../results/timing/{model_name}/{model_name}-{key}-{definition}
                                                        -{current_time}.csv",
                "w",
                newline="",
            ) as file:
                writer = csv.writer(file)
                writer.writerow(["model_name", model_name])
                writer.writerow(["prompt_key", key])
                writer.writerow(["definition", definition])
                writer.writerow(["time_taken", time.time() - startTime])
                writer.writerow(["timestamp", current_time])
                writer.writerow(["endpoint_name", endpoint_name])
                writer.writerow(["parameters", parameters])

        process_csv(
            f"../results/raw/{model_name}/{filename}.csv",
            f"../results/clean/{model_name}/{filename}-clean.csv",
            model_family == "mistral",
        )
```

A listagem C.2 é o código utilizado para avaliação dos modelos disponíveis através da plataforma OpenAI.

Listagem C.2: Código responsável pela avaliação dos modelos provenientes da OpenAI e cálculos das métricas de análise.

```
import pandas as pd
from sklearn.preprocessing import label_binarize
from sklearn.metrics import (
    classification_report,
    confusion_matrix,
    ConfusionMatrixDisplay,
    matthews_corrcoef,
```

```
)
import csv
from transformers import GPT2Tokenizer
from datasets import load_from_disk
import openai
from tqdm import tqdm
import time
from requests.exceptions import HTTPError
import os

import numpy as np
import time
from utils import (
    generate_instruction_prompt,
    prepare_prompt_list,
    process_csv,
    get_generated_text,
)

model_name = "gpt-4-0125-preview"
prompt = """
    Classify the following Dota 2 chat message as 'Toxic' or 'Not-Toxic'.
    """


tokenizer = GPT2Tokenizer.from_pretrained("gpt2")
test_dataset = load_from_disk("../dataset/test")
# test_dataset = load_from_disk("../dataset/test").select(range(3))

ai_message_template = {"role": "system", "content": prompt}


def classify_question_type(question: str) -> str:
    ai_response = ""
    messages = [ai_message_template, {"role": "user", "content": question}]
    try:
        response = openai.chat.completions.create(
            model=model_name,
            messages=messages,
            temperature=0.0,
            max_tokens=512,
        )

        generated_text = response.choices[0].message.content
    except openai.APIError as e:
        # Handle API error here, e.g. retry or log
        print(f"OpenAI API returned an API Error: {e}")
        pass
    except openai.error.APIConnectionError as e:
        # Handle connection error here
        print(f"Failed to connect to OpenAI API: {e}")
        pass
    except openai.error.RateLimitError as e:
        # Handle rate limit error (we recommend using exponential backoff)
        print(f"OpenAI API request exceeded rate limit: {e}")
        pass

    ai_response = response.choices[0].message.content
    generated_text = response.choices[0].message.content
```

```
67
68      input_tokens = len(tokenizer.encode(" ".join([str(x) for x in messages])))
69      output_tokens = len(tokenizer.encode(ai_response))
70
71      return ai_response, input_tokens, output_tokens
72
73
74  # Run the prediction on the dataset and save as a results.csv file
75
76
77  def run_prediction():
78      current_time = time.strftime("%Y-%m-%dT%H:%M:%S")
79      inputs = [str(sample["context"]) for sample in test_dataset]
80      true_labels = [str(sample["completion"]) for sample in test_dataset]
81
82      # Load existing results if they exist
83      # if os.path.exists(output_path):
84      #     results_df = pd.read_csv(output_path, sep=";")
85      # else:
86      #     reference, prediction, input_tokens, output_tokens
87      results_df = pd.DataFrame(
88          columns=["reference", "prediction", "input_tokens", "output_tokens"]
89      )
90
91      filename = f"predictions-{model_name}-zero-shot-none-{current_time}"
92      startTime = time.time()
93      for input, true_label in tqdm(zip(inputs, true_labels), total=len(inputs)):
94          prediction, input_tokens, output_tokens = classify_question_type(input)
95
96          results_df = results_df._append(
97              {
98                  "reference": true_label,
99                  "prediction": prediction,
100                 "input_tokens": input_tokens,
101                 "output_tokens": output_tokens,
102             },
103             ignore_index=True,
104         )
105
106     results_df.to_csv(f"../results/raw/{model_name}/{filename}.csv", index=False)
107
108     with open(
109 f"../results/timing/{model_name}/{model_name}-zero-shot-none-{current_time}.csv",
110         "w",
111         newline="",
112     ) as file:
113         writer = csv.writer(file)
114         writer.writerow(["model_name", model_name])
115         writer.writerow(["prompt_key", "zero-shot"])
116         writer.writerow(["definition", "none"])
117         writer.writerow(["time_taken", time.time() - startTime])
118         writer.writerow(["timestamp", current_time])
119         writer.writerow(["endpoint_name", "openAI"])
120         writer.writerow(["parameters", ""])
121
122     process_csv(
123         f"../results/raw/{model_name}/{filename}.csv",
124         f"../results/clean/{model_name}/{filename}-clean.csv",
```

```
125             model_name == "gpt",
126         )
127
128
129 run_prediction()
```

O código completo está disponível no GitHub[1].

[1] https://github.com/opauloxavier/llm-dissertation