
Felipe Jun Ting Lin

# A UTILIZAÇÃO DE MODELOS DE LLM PARA GERAÇÃO DE BULAS AUTOMATIZADAS

Trabalho de Graduação

Universidade Federal de Pernambuco
graduacao@cin.ufpe.br
portal.cin.ufpe.br/graduacao

RECIFE
2023

Universidade Federal de Pernambuco
Centro de Informática
Graduação em Engenharia de Computação

Felipe Jun Ting Lin

# A UTILIZAÇÃO DE MODELOS DE LLM PARA GERAÇÃO DE BULAS AUTOMATIZADAS

*Trabalho apresentado ao Programa de Graduação em Engenharia de Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Engenharia de Computação.*

*Orientador: Prof. Dr. Luciano de Andrade Barbosa*

RECIFE
2023

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

Lin, Felipe Jun Ting.
A utilização de modelos de LLM para geração de bulas automatizadas / Felipe Jun Ting Lin. - Recife, 2023.
56 p. : il., tab.

Orientador(a): Luciano de Andrade Barbosa
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Informática, Engenharia da Computação - Bacharelado, 2023.

1. Processamento de Linguagem Natural. 2. Large Language Models. 3. Inteligência Artificial. 4. Chatbot. 5. Automedicação. I. Barbosa, Luciano de Andrade. (Orientação). II. Título.

600 CDD (22.ed.)

Tese de graduação apresentada por **Felipe Jun Ting Lin** ao programa Graduação em Engenharia da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título **A utilização de modelos de LLM para geração de bulas automatizadas**, orientada pelo **Prof. Dr. Luciano de Andrade Barbosa** e aprovada pela banca examinadora formada pelos professores:

---

Prof. Tsang Ing Ren
Centro de Informática/UFPE

RECIFE
2023

*Dedico este trabalho para minha família, amigos,*
*professores e colegas de trabalho que sempre me apoiaram*
*ao longo dessa jornada.*

# Agradecimentos

Gostaria de utilizar esse espaço para expressar meus sinceros agradecimentos a todos que fizeram parte dessa minha jornada durante esse tempo na faculdade e tornou possível a realização deste trabalho de graduação.

Agradeço à minha família por ter me dado apoio e incentivos que foram fundamentais, desde o início, ao longo dessa trajetória para que eu pudesse seguir essa minha paixão pela área de tecnologia.

Aos meus amigos da faculdade, que foram companheiros desde o primeiro dia de aula, sempre unidos, seja nos desafios das provas, nas realizações de trabalhos em equipe e até mesmo nas atividades extracurriculares.

Ao professor Luciano de Andrade, sou muito grato pela orientação e pelo constante apoio. Sempre dedicado em esclarecer minhas dúvidas e as respostas foram fundamentais para a conclusão deste trabalho. Também estendo meu agradecimento aos demais professores que tive a honra de ser lecionado, contribuindo para que eu tivesse um conhecimento sólido na área de computação.

Aos meus colegas de trabalho, minha admiração pela paciência e pela colaboração que encontrei em vocês. Ter uma jornada profissional ao lado de pessoas tão competentes e comprometidas me fez com que eu pudesse amadurecer e desenvolver habilidades que hoje considero essenciais.

Ao Centro de Informática e à Universidade Federal de Pernambuco pela notável estrutura proporcionada ao longo destes cinco anos de graduação. O ambiente repleto de mentes brilhantes certamente impulsionou meu crescimento pessoal e profissional e me sinto privilegiado por ter tido oportunidade de fazer parte disso.

Por fim, agradeço a todos que, de alguma forma, participaram e contribuíram nessa minha jornada. Cada interação, conversa e momento compartilhado deixou uma impressão marcante na minha formação. E estou ansioso para aplicar os conhecimentos adquiridos e retribuir, de alguma forma, tudo o que recebi.

*Success depends upon previous preparation, and without such preparation there is sure to be failure.*

*—CONFUCIUS*

# Resumo

Conforme o avanço da área de medicina, surgiram mais opções de tratamentos com uso de medicamentos e baseado nisso, torna-se natural que as pessoas procurem informações sobre esses remédios para entender o seu funcionamento e tomar decisões informadas sobre sua saúde. Dessa forma, esse trabalho impactará diretamente nos pacientes e nos profissionais da saúde ao utilizar a abordagem de modelos de linguagem baseados em aprendizado de máquina, mais especificamente o LLM (*Large Language Model*) a fim de facilitar a busca em bulas de medicamentos com uma linguagem humana. Ao digitar o nome do remédio e escrever a finalidade da pergunta que deseja fazer ao sistema, o modelo LLM tem o objetivo de gerar resumos, tirando dúvidas do usuário e respondendo de forma prática.

**Palavras-chave:** Processo de Linguagem Natural, Modelo de Linguagem Grande, Geração de Texto, Respostas Automatizadas Sobre Remédios

# Abstract

As the field of medicine advances, more options for treatments with the use of medication, and based on that, it becomes natural for people to look for information about these drugs to understand how they work and make informed decisions about their health. That said, this work will directly impact patients and healthcare professionals by using the approach of language models based on machine learning, more specifically the LLM (Large Language Model) in order to facilitate the search in leaflets of medicines with a human language. By typing the name of the medicine and writing the purpose of the question you want to ask the system, the LLM model aims to generate summaries, taking user questions and answering in a practical way.

**Keywords:** Natural Language Processing, Large Language Model, Text Generation, Automated Answers About Remedies

# Lista de Figuras

3.1 Processo de Scraping . . . . . 28
3.2 Arquitetura do *Transformers* . . . . . 30

4.1 Fluxograma do Desenvolvimento do Projeto . . . . . 40
4.2 *Leaderboard* dos Modelos 7B do HuggingFace . . . . . 43

5.1 *Boxplot* do *BLEU Score* de Todos Modelos . . . . . 48
5.2 *Boxplot* do *Rouge-L Score* de Todos Modelos . . . . . 48

# Lista de Tabelas


4.1 Exemplo de Base de Dados com Perguntas e Contexto usando *embedding* do *OpenAI* . . . . . 42
4.2 Exemplo de Base de Dados com Perguntas e Contexto usando *embedding* do *HuggingFace* . . . . . 42
4.3 Exemplo de Respostas Geradas pelos Modelos utilizando *embedding* do *OpenAI* 44
4.4 Exemplo de Respostas Geradas pelos Modelos utilizando *embedding* do *HuggingFace* . . . . . 44

5.1 Exemplo da Tabela de *Ground_Truth* . . . . . 46
5.2 Resultado dos Modelos Utilizando *Embedding* do *OpenAI* . . . . . 46
5.3 Resultado dos Modelos Utilizando *Embedding* do *HuggingFace* . . . . . 46
5.4 Exemplo de Melhor Resultado Usando *Embedding* do *OpenAI* no Modelo *Falcon-7b* . . . . . 49
5.5 Exemplo de Melhor Resultado Usando *Embedding* do *Multi-QA* no Modelo *Falcon-7b* . . . . . 49

# Lista de Acrônimos


**ANN** *Aproximate Nearest Neighbors* . . . . . 29

**PLN** Processamento de Linguagem Natural . . . . . 29

**IA** Inteligência Artificial . . . . . 25

**RNN** *Recurrent Neural Network* . . . . . 30

**LLM** *Large Language Models* . . . . . 23

**GPT** *Generative Pre-Trained Transformers* . . . . . 31

**ML** *Machine Learning* . . . . . 34

**CSV** *Comma-Separated Values* . . . . . 40

**FAISS** *Facebook AI Similarity Search* . . . . . 40

**RAG** *Retrieval Augmented Generation* . . . . . 33

**CFF** Conselho Federal de Farmácia . . . . . 23

**ICTQ** Instituto de Ciência, Tecnologia e Qualidade . . . . . 23

**LCS** *Longest Common Subsequence* . . . . . 47

# Sumário


**1 Introdução** **23**

1.1 Contexto . . . 23

1.2 Propósito e Relevância Deste Trabalho . . . 23

**2 Justificativa** **25**

**3 Fundamentação** **27**

3.1 Obtenção de dados . . . 27

3.1.1 *Scraping* . . . 27

3.2 Pré-Processamento de Texto . . . 28

3.2.1 *Chunking* . . . 28

3.2.2 *Embedding* . . . 29

3.3 Bancos de Dados de Vetores . . . 29

3.4 Modelos de Linguagens Pré-Treinados . . . 29

3.4.1 *Transformers* . . . 30

3.5 *Large Language Models* . . . 31

3.5.1 *Prompt Engineering* . . . 31

3.5.2 GPT . . . 32

3.5.3 Falcon . . . 32

3.5.4 LLaMA2 . . . 33

3.6 RAG . . . 33

3.6.1 Definição . . . 33

3.6.2 Possíveis Aplicações . . . 34

3.7 Ferramentas de Código Aberto . . . 34

3.7.1 *HuggingFace* . . . 34

3.7.2 LangChain . . . 35

3.8 Métricas de Avaliação . . . 35

3.8.1 *BLEU Score* . . . 35

3.8.2 *ROUGE* . . . 36

**4 Metodologia e Desenvolvimento** **39**

4.1 Extração e Processamento dos Dados . . . 39

4.2 Transformação de Dados em Vetores . . . 40

4.3 Desenvolvimento de Prompt . . . 41

4.4 Modelos Selecionados . . . . . 42
4.5 Geração de Resposta . . . . . 43

**5 Avaliação dos Modelos** **45**

**6 Conclusão** **51**

**Referências** **53**

# 1 Introdução

## 1.1 Contexto

No cenário de saúde pública, a prática de automedicação representa uma questão de grande relevância. De acordo com os dados obtidos pelo Conselho Federal de Farmácia (CFF), foi revelado que, em 2019, cerca de 77% das pessoas no Brasil tinham o hábito de se automedicar [8]. Apesar dessa prática parecer uma solução conveniente para aliviar sintomas mais leves, representa um risco à saúde pública, uma vez que pode resultar em sérios problemas de saúde decorrente da falta de conhecimento sobre a composição de um determinado remédio, resultando no agravamento de condições médicas. Existe um outro estudo, levantado pelo Instituto de Ciência, Tecnologia e Qualidade (ICTQ) que mostra que o número de pessoas que se automedicam vem aumentando substancialmente desde o ano de 2014, em que os principais medicamentos utilizados são: analgésicos - 64%, antigripais - 47%, relaxantes musculares - 35% e controle de ansiedade, estresse e insônia - 6% [20].

Esses dados levantam uma preocupação fundamental visto que isso indica não apenas uma falta de conscientização sobre os riscos da automedicação, mas também aponta uma urgência em realizar uma educação em saúde e medidas preventivas. O ato de automedicação pode mascarar alguns sintomas graves que possam existir, atrasando futuros diagnósticos adequados e contribuir para o uso impróprio de medicamentos.

## 1.2 Propósito e Relevância Deste Trabalho

Em um cenário em que a automedicação é uma preocupação crescente, este trabalho propõe uma solução inovadora capaz de instruir os usuários e diminuir os riscos presentes para esse tipo de prática. O objetivo é desenvolver uma aplicação baseada em *Large Language Models* (LLM) com finalidade de responder às perguntas dos usuários sobre medicamentos, gerando informações essenciais, como: efeitos colaterais, contraindicações, composição, posologia e

outros dados relevantes para que possam tirar dúvidas dos usuários.

Ao longo do projeto, será mostrado as etapas para a sua implementação que pode ser tanto em modelos de código aberto, como *falcon* e *llama*, quanto em modelos de código fechado, como o *gpt-3.5-turbo*. Essa abordagem tem como objetivo permitir uma análise comparativa dos desempenhos desses modelos, produzindo resultados que buscam fornecer informações seguras e coerentes em um cenário de automedicação em crescimento. Portanto, essa análise se torna fundamental para avaliar a eficácia da aplicação do projeto e uma possível influência que pode exercer na promoção de práticas de saúde mais responsáveis e conscientes.

# 2

# Justificativa

Com o surgimento da área de Inteligência Artificial (IA), sua evolução tem sido bastante notável. Desde a sua origem, quando a IA mal conseguia resolver problemas simples, como o "ou exclusivo"(XOR) [26], para dias atuais, em que é possível automatizar processos e criação de aplicações independentes capazes de solucionar uma variedade de desafios. Esse avanço mostrou inúmeras possibilidades que IA pode incorporar para resolver problemas complexos dos usuários, [13], englobando desde sistemas de reconhecimento facial, sistemas de recomendação, veículos autônomos, até *chatbots* que será abordado de forma mais elaborado ao longo do projeto.

Em 2022, a empresa *OpenAI* introduziu uma ferramenta revolucionária chamada de *ChatGPT*, que ganhou rápida popularidade devido à sua facilidade de uso e na sua capacidade de gerar soluções coerentes aos usuários, utilizando uma linguagem próxima à compreensão humana [42]. O *ChatGPT* demonstrou que é capaz de responder a uma ampla variedade de perguntas e tópicos, como resumir textos, escrever códigos, fornecer informações sobre diversos assuntos e muito mais, se tornando uma ferramenta de busca eficiente, visto que consegue ajudar a poupar tempo e na maioria das vezes traz consigo uma resposta satisfatória [18].

No entanto, embora o *ChatGPT* possa parecer como uma solução sólida, ele ainda apresenta algumas limitações. O modelo utilizado, *GPT-3.5-Turbo*, o qual é disponibilizado gratuitamente para o público, possui algumas restrições quanto ao número de *tokens* que os usuários podem realizar em uma consulta e na quantidade de *tokens* dos resultados gerados. Uma outra desvantagem desse modelo é que nem sempre ele oferece respostas corretas, uma vez que ele foi projetado para criar respostas e não se preocupar com sua corretude, sendo necessário a checagem da fonte após a obtenção dos resultados através dessa aplicação [11].

Diante das considerações citadas acima, surge a justificativa para o uso de LLM, modelos como *falcon*, *llama* e até mesmo o *gpt-3.5-turbo* como uma alternativa eficaz ao abordar as limitações associadas às soluções tradicionais. Os LLM são capazes de compreender e gerar texto em linguagem natural e oferece também a flexibilidade e controle nas aplicações que serão implementadas, podendo personalizar nas respostas que vão ser geradas e torná-las ideais para a interação com os usuários.

Dessa forma, como o objetivo desse projeto é desenvolver uma aplicação para fornecer informações detalhadas e confiáveis sobre medicamentos, o uso de LLM se torna ideal para esse contexto. A partir dos resultados gerados, o usuário é capaz de obter conhecimentos e informações precisas, auxiliando na tomada de decisões responsáveis em relação ao uso de um determinado medicamento com propósito de melhorar e proteger a saúde da população em geral, como mencionado em Capítulo 1.

# 3

# Fundamentação

O objetivo deste capítulo é de proporcionar uma base sólida para o entendimento abrangente desse trabalho, onde serão abordados diversos termos e conceitos essenciais, no qual, cada um desses tópicos teve um papel essencial na realização e na análise do projeto. Após a explicação detalhada desses elementos, os leitores estarão aptos para compreender a metodologia, os resultados e as implicações desse trabalho de forma mais aprofundada.

## 3.1 Obtenção de dados

A obtenção de dados foi através desse artigo *Leaflet of Medicine Corpus* [34], onde foi disponibilizado na plataforma de *github* 1050 arquivos, cada um repleto de informações cruciais sobre um remédio específico, fornecendo os *insights* necessários para as análises ao longo do projeto.

### 3.1.1 *Scraping*

A técnica de *scraping*, mostrado na Figura 3.1, consiste em extrair informações de forma automatizada de um ambiente digital repleto de dados de maneira sistemática e eficiente [24]. Após a extração de dados, as informações são convertidas em conteúdos em um formato acessível e pronto para análises posteriores. Numa era onde a internet desempenha um papel crucial como repositório de informações, essa técnica se torna um artefato indispensável para os pesquisadores e profissionais que buscam explorar e compreender os dados disponíveis.

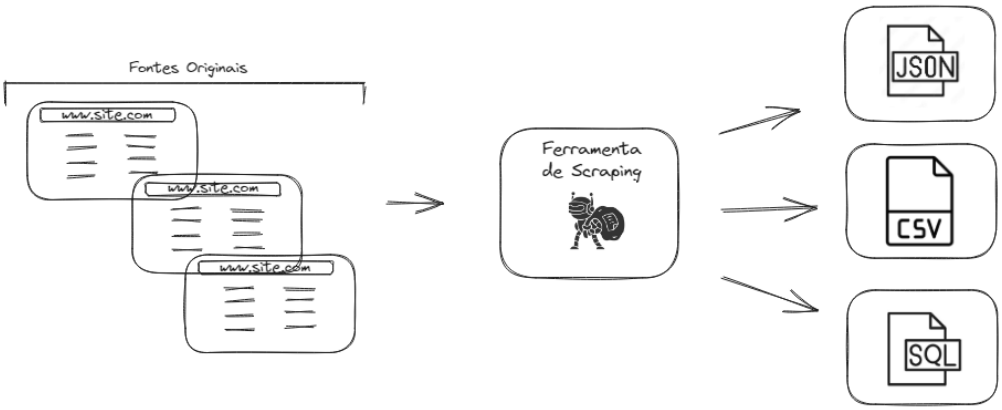


**Figura 3.1:** Processo de Scraping

## 3.2 Pré-Processamento de Texto

No cenário da análise de dados e processamento de linguagem natural, a técnica de pré-processamento de texto, surge como uma etapa primordial para a limpeza de dados e remoção de ruídos antes que os dados sejam aplicados pelos algoritmos e modelos de aprendizado de máquina [43, 4]. Ao realizar esse procedimento nos textos, é possível garantir uma melhora na qualidade e precisão de qualquer análise no decorrer do projeto. Além disso, ao organizar os dados textuais em um formato uniforme e fácil de compreender, o pré-processamento permite que os modelos de aprendizado de máquina possam aprender padrões de forma mais eficiente.

### 3.2.1 *Chunking*

A técnica de *chunking* tem um papel indispensável para decompor e compreender de maneira mais eficaz as unidades semânticas contidas nos textos [31]. Ela consiste em segmentar o texto em parte menores e mais coesas, onde cada um desses pedaços representam uma unidade semântica que pode consister em uma ou mais palavras, formando blocos de significados mais coerentes [41].

Um dos principais motivos para realizar o *chunking* de textos é de facilitar a análise semântica e na extração de informações, uma vez que em textos mais extensos, as informações podem estar dispersas em várias partes do texto. Dessa forma, a técnica auxilia na identificação das palavras que estão relacionadas e na identificação das partes do texto que são relevantes à análise. Sendo assim, ao dividir um texto em pedaços significativos, é possível capturar as relações entre palavras e conceitos, dando possibilidade para realizar uma análise mais precisa e profunda.

### 3.2.2 *Embedding*

A técnica de *embedding* consiste em mapear as palavras individuais em forma de números reais e armazenadas em um espaço vetorial, no qual, cada palavra pode ser representada por um vetor de valor real com dezenas ou até centenas de dimensões [33, 23]. Dessa forma, ao realizar esse mapeamento, é possível determinar as distâncias e observar as relações semânticas entre cada um desses vetores [12]. Isso significa que palavras que são semanticamente semelhantes são mapeadas para vetores próximos, enquanto palavras com significados diferentes são mapeadas para vetores distantes, permitindo que algoritmos de aprendizado de máquina possam entender as nuances e os contextos presentes de uma linguagem humana.

## 3.3 Bancos de Dados de Vetores

Um banco de dados vetoriais é uma estrutura de armazenamento que lida especificamente com vetores numéricos, permitindo que os usuários façam buscas por meio de similaridade de vetores. Esses vetores podem representar várias formas de informações, desde representações de palavras até características de objetos em análises de imagem [39]. Desse modo, esse tipo de banco de dados é bastante utilizado em mecanismos de busca, no qual, os vetores são criados a partir de representações numéricas de frases, documentos ou outros itens indexados para a pesquisa.

O motivo dos bancos de dados vetoriais conseguirem performar em alta velocidade é devido a utilização do algoritmo de *Aproximate Nearest Neighbors* (ANN) [10], no qual tem o objetivo de encontrar, de forma aproximada, o ponto mais próximo a um determinado ponto de consulta em um conjunto de pontos em um espaço métrico. Em outros termos, ao invés de tentar encontrar o ponto exato mais próximo, esse algoritmo tem o objetivo de encontrar um ponto que esteja próximo o suficiente em relação ao ponto de consulta [1].

Portanto, o objetivo principal para a utilização de bancos de dados de vetor é a necessidade de gerenciar e realizar consultas sobre dados complexos de maneira eficaz. Ao lidar com grandes volumes de informações, por exemplo: nas áreas de Processamento de Linguagem Natural (PLN) ou de análise de imagens, as representações numéricas de dados são essenciais e os vetores têm o papel fundamental de fazer com que essas informações e contextos sejam convertidas de tal forma que os algoritmos sejam capazes de compreender e processar.

## 3.4 Modelos de Linguagens Pré-Treinados

A ideia de utilizar modelos de IA pré treinados está relacionado ao conceito de utilizar a aprendizagem por transferência [44], em que é usado as informações que eram projetados

para realizar uma tarefa específica e reutiliza o modelo para executar outra atividade que pode ser de uma área correlacionada [38]. Dessa forma, o modelo possui conhecimentos prévios de uma estrutura da linguagem e consegue capturar padrões semânticos e gramaticais existentes em outros textos.

### 3.4.1 *Transformers*

*Transformers* é uma arquitetura de rede neural que visa resolver as tarefas que possuem uma sequência de textos. Diferente de alguns modelos baseados em *Recurrent Neural Network* (RNN), os *Transformers* não dependem de conexões sequenciais e são capazes de capturar o contexto de longo prazo em um texto.

A arquitetura do *Transformers*, mostrado na Figura 3.2, é composta por dois componentes, o *encoder* que é responsável por processar entrada e o *decoder* que é responsável para gerar saídas [37]. Um exemplo que pode ser empregado o uso de *Transformers* é no serviço de tradução, em que o texto original pode ser considerado como sequências de valores da entrada e o texto convertido como sequências de valores da saída.

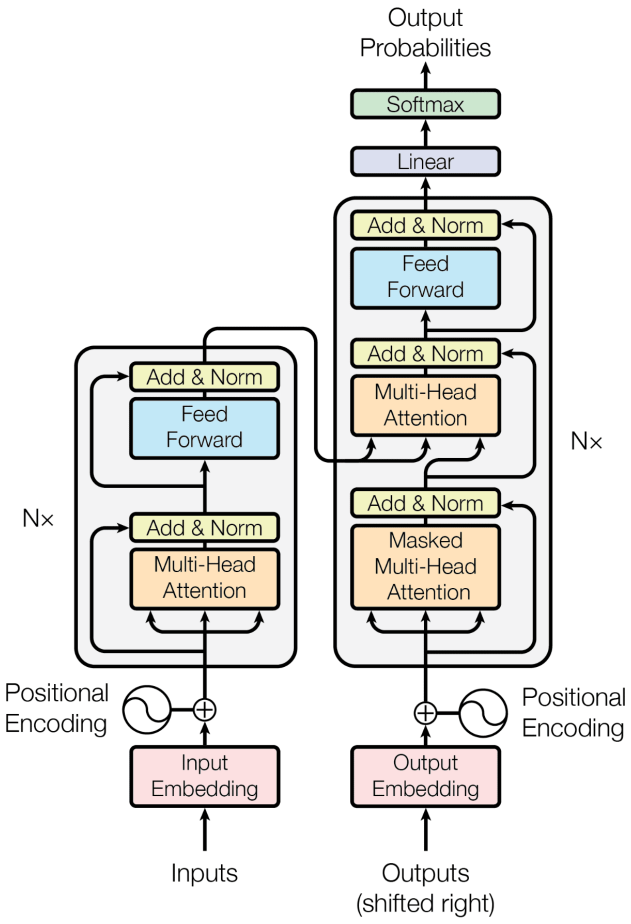


**Figura 3.2:** Arquitetura do *Transformers*

## 3.5 *Large Language Models*

Os LLM são modelos de aprendizado de máquina que podem executar diversas tarefas de PLN, como geração de textos automatizadas, responder perguntas do usuário (*chatbots*), traduzir textos, etc...[17]. Geralmente as suas implementações são baseadas na arquitetura de *transformers* [7, 37], ou seja, ao contrário de RNN que usam recorrência como estratégia principal para determinar relacionamentos entre tokens de uma sequência, as redes *transformers* usam o mecanismo de *self-attention* para determinar esses relacionamentos e a partir de soma ponderada de uma sequência específica, o modelo determina dinamicamente quais tokens na sequência são mais relevantes entre si.

Como o nome sugere, os LLM são modelos caracterizados por sua grandeza e capacidade de processar grandes volumes de texto. Dessa forma, eles são capazes de entender e gerar uma resposta legível na linguagem humana, tornando-os aptos a serem aplicados em diferentes domínios [30]. Seus resultados eficientes advêm da alta capacidade de gerar uma saída com uma interpretabilidade satisfatória devido a sua arquitetura robusta ter sido treinada por meio de aprendizado supervisionado sobre um extenso conjunto de dados textuais. Atualmente os modelos mais prestigiados, considerados o estado da arte, possuem dezenas de bilhões de parâmetros (variáveis ajustáveis no treinamento), sendo eles o modelo de *Generative Pre-Trained Transformers* (GPT) [30], *LLaMA* [36] e *Falcon* [29] que serão abordados ao longo do projeto.

### 3.5.1 *Prompt Engineering*

O *prompt engineering* é uma técnica que se refere a formular *prompts*, ou seja, instruções específicas fornecidas aos modelos de linguagem com o propósito de ter as respostas de forma desejada [40]. Essa é uma etapa crucial no uso de modelos de linguagem, como, por exemplo, nas tarefas de geração de texto, em que os modelos de IA geram respostas automáticas com base nas entradas recebidas. Sendo assim, os *prompts* são importantes para que a qualidade e a relevância das respostas obtidas através dos modelos de linguagem sejam coesas, atendendo às expectativas do usuário.

Ao fornecer na entrada essas instruções específicas, os desenvolvedores têm a capacidade de moldar as respostas, permitindo a adaptação dos modelos a cenários específicos. Então, mesmo que os modelos de IA possam ter sido treinados previamente com grandes volumes de dados, eles ainda podem gerar respostas que não se adequam perfeitamente ao contexto ou na linguagem humana. Dessa forma, ao invés de confiar apenas no treinamento que os modelos de IA receberam anteriormente, através dos *prompts*, as interações com o modelo de IA podem ser customizados para situações específicas, garantindo que as respostas geradas sejam mais precisas e alinhadas com as necessidades específicas do usuário, sem ter que refazer ou reformular a pergunta várias vezes.

### 3.5.2 GPT

Os modelos de GPT, desenvolvido por *OpenAI*, são um conjunto de rede neural que utilizam a arquitetura de *transformers* usados para tarefas de processamento de linguagem natural e oferecem soluções e capacidade de gerar conteúdo similar aos feitos humanos ao gerar uma resposta legível e coerente [30]. Esses modelos são pré-treinados em grandes quantidades de dados, a partir das técnicas de *scraping* e de outras fontes de conteúdos, como por exemplo *Wikipedia* ou outros *sites* que podem ser encontrados na *Internet*.

No entanto, a grande vantagem do modelos de GPT se deve ao fato deles serem um modelo pré-treinado, isso significa que o modelo é treinado com grande volume de textos em linguagem natural e é capaz de ajustar ele para executar algumas tarefas específicas posteriormente [30]. Dessa forma, isso permite que o GPT possa ter um entendimento da gramática e do contexto da linguagem humana.

Até o momento da publicação desse texto, existem quatro versões de modelos GPT existentes: o *GPT-1* foi lançado em 2018 por *OpenAI* e tinha cerca de 117 milhões de parâmetros e na época ele já tinha uma grande capacidade de gerar textos coerentes dado um certo contexto a ele, entretanto, possuía algumas limitações por gerar textos repetitivos quando o contexto dado não existe durante o seu treinamento. O modelo *GPT-2*, lançado em 2019, já era consideravelmente maior ao comparar com o seu antecessor, possuía 1.5 bilhões de parâmetros e era capaz de gerar resultados melhores e, também, ao conter maior quantidade de conteúdo durante o seu treinamento, o tornava apto de trazer um resultado melhor, mas ainda assim, não conseguia trazer a mesma excelência ao gerar textos grandes. Logo, o modelo *GPT-3*, com seus 175 bilhões de parâmetros [3] foi treinado com um banco de dados que beira cerca de trilhões de palavras, permitindo que ele consiga gerar resultados mesmo que o usuário não tenha fornecido contexto a ele, mas isso demonstra que esse modelo pode gerar respostas enviesadas e inapropriadas. Por fim, o modelo *GPT-4*, lançado em 2023 é capaz de contornar algumas dificuldades que os modelos antecessores tinham e tornou possível a inserção de entradas utilizando imagens.

### 3.5.3 Falcon

Falcon é o modelo de código aberto que foi desenvolvido por *Technology Innovation Institute (TII)* e é capaz de executar tarefas de processamento de texto como por exemplo, criar conteúdos, resolver problemas complexos, ser um assistente virtual, tradução e análise de sentimento. A variação que possui melhor desempenho é o modelo *Falcon-40B* que, como o nome sugere, possui 40 bilhões de parâmetros e foi treinado por 1 trilhão de *token*s no banco de dados chamado *RefinedWeb*, criado pela mesma equipe. Por ser o modelo de código aberto, permite que ele seja de uso comercial e dando oportunidade para que qualquer usuário possa utilizar o modelo e construir suas aplicações com respostas eficientes [29].

Embora o modelo Falcon possa fornecer respostas relevantes e coerentes em muitos contextos, devido ao fato de ter aprendido grandes quantidades de texto da *internet*, pode gerar ocasionalmente respostas que são irrelevantes, enviesadas ou fora de contexto. No entanto, o fato de ser um modelo de código aberto permite que a comunidade de desenvolvedores e usuários possam ter a possibilidade de melhorar e aprimorar o modelo continuamente, permitindo corrigir essas falhas existentes e trazendo mais melhorias às respostas geradas.

### 3.5.4 LLaMA2

LLaMA2 é o modelo generativo desenvolvido pela equipe do *Meta* baseado na arquitetura original dos *transformers* e foi treinado através de um banco de dados que é cerca de 40% vezes maior que nos modelos anteriores. Ele possui 3 variantes, LLaMA-7b, LLaMA-13b e LLaMA-70b, cada um com respectivamente 7, 13, 70 bilhões de parâmetros e possui uma grande capacidade e eficiência em gerar respostas com linguagem compreensiva aos humanos [36]. Logo, por ser um modelo pré-treinado, ele é capaz de gerar resultados utilizando linguagem natural e, caso o usuário faça uma pergunta sob um contexto que o modelo não tenha o conhecimento, é possível aplicar a técnica de *fine-tuning* e, através de técnicas de *prompt engineering*, é provável de obter uma resposta desejada.

## 3.6 RAG

### 3.6.1 Definição

O *Retrieval Augmented Generation* (RAG) é uma técnica utilizada na área de PLN que utiliza da estratégia de recuperação de dados (*retrieval*) e geração de texto (*generation*) [15]. O propósito dele é melhorar a qualidade e relevância das respostas geradas por modelos de linguagem [25]. Adicionalmente, essa técnica permite que um modelo consiga ter acesso às informações de uma base de conhecimento e, dentre desse contexto, é possível gerar respostas mais precisas e informativas que são alinhadas com as referências originais para as perguntas dos usuários, trazendo os resultados mais significativos [19].

Como dito anteriormente, a primeira parte do RAG é sobre a recuperação de informações relevantes a partir de um banco de dados. Essa etapa é feita através de técnicas de busca e recuperação de informações que tem como objetivo identificar algumas similaridades presentes no texto que possam conter informações expressivas.

A segunda parte do RAG é sobre a etapa de geração de respostas com base nas informações recuperadas na etapa anterior. Para isto, existem diversos LLM, como por exemplo: *GPT-3*, *Falcon-7b*... que são regularmente utilizados nessa fase para criar respostas coerentes com base no contexto obtido.

### 3.6.2 Possíveis Aplicações

Como o campo de PLN é bastante amplo, podendo atuar em diversas áreas, a técnica de RAG também se tornou bastante eficaz em diversas aplicações [21], trazendo várias vantagens nos domínios de, por exemplo:

1. **Aprimoramento da Qualidade das Respostas**: Com a utilização de contexto recuperados a partir de referências originais, o RAG consegue entregar respostas relevantes e contextualmente apropriadas, de forma que os resultados são capazes de se encaixar no âmbito da pergunta.

2. **Responder a Perguntas Complexas**: Visto que nem sempre o usuário consegue deixar claro as suas dúvidas, o uso de RAG consegue lidar bem com esse tipo de situação, pois permite que o modelo consiga acessar informações relevantes a partir de uma base de dados, capacitando o LLM a responder de maneira precisa e coerente.

3. **Diversidade de Usos**: Além de responder a perguntas, o RAG tem sido utilizado em tarefas como tradução, resumir os textos, criar *chatbots*... Sua capacidade de melhorar a qualidade das respostas é valiosa em uma ampla variedade de cenários.

Sendo assim, o RAG surgiu como uma solução promissora para superar as limitações de LLM. Ao utilizar essa técnica, os desenvolvedores conseguem obter soluções personalizadas, mantendo a coerência dos dados, ao passo de utilizar ao máximo as capacidades dos LLM em gerar resultados [32]. E, com o passar do tempo, conforme forem surgindo mais pesquisas nessa área, é possível ter ainda modelos de linguagem mais robustos e eficazes no futuro.

## 3.7 Ferramentas de Código Aberto

### 3.7.1 *HuggingFace*

*HuggingFace* é uma plataforma que permite com que os usuários possam criar, treinar, utilizar os modelos de *Machine Learning* (ML) de código aberto e colaborar com outros entusiastas da área sendo capaz de desenvolver aplicações de ponta a ponta.

Um dos pontos fortes dessa plataforma é a criação e a manutenção de bibliotecas como *transformers*, pois permite que as pessoas possam ter um fácil acesso a ampla variedade de modelos de linguagem de grande escala, como GPT, *Falcon*, LLaMA, entre outros. Essas bibliotecas simplificam a implementação de tarefas como geração de texto, classificação de sentimento, tradução automática e muito mais, tornando a IA mais acessível para desenvolvedores de todos os níveis de experiência, economizando tempo e recursos.

### 3.7.2 LangChain

*LangChain* é um projeto de código aberto criado por Harrison Chase para facilitar a implementação de aplicativos baseados em LLM, permitindo que seja possível aproveitar as tecnologias emergentes advindas do PLN.

Essa ferramenta agiliza o desenvolvimento de diversos tipos de aplicações, tais como as atividades de *chatbots*, geração de perguntas e respostas automatizadas e realizar resumos sobre um determinado material. Ela basicamente realiza o "encadeamento"de componentes de vários módulos, permitindo a criação de aplicações construídas em torno de um LLM.

## 3.8 Métricas de Avaliação

A avaliação de modelos é uma parte essencial no desenvolvimento de qualquer aplicação que utiliza de IA como base e isso também inclui nos modelos de LLM. Para isso, durante o projeto foi utilizado duas métricas: *BLEU Score* [28] e *ROUGE* [22].

### 3.8.1 *BLEU Score*

O *BLEU Score*, cujo acrônimo em inglês é *Bilingual Evaluation Understudy*, é uma métrica usada para avaliar a qualidade de resposta gerada por modelos de linguagens. Foi inicialmente desenvolvido para avaliar traduções, mas também é bastante aplicado em tarefas mais amplas de PLN [28].

Essa métrica tem um valor entre 0 e 1 e, quanto maior o resultado, maior é a correspondência entre a resposta gerada e a referência original. O *BLEU Score* calcula a sobreposição de N-gramas entre o resultado gerado e o texto original. Cada N-grama é uma sequência de N *tokens* consecutivos que, geralmente, os valores usados são 1,2,3 e 4, representando unigrama, bigramas, trigramas e quadrigrama, onde o *token* é uma unidade de texto, podendo ser uma palavra ou uma sílaba da mesma. Para cada N-grama, o *BLEU* observa quantos esses *tokens* geradas pelo resultado estão presentes nos textos originais, depois é realizada a contagem de N-gramas corretas para calcular a sua precisão. Por fim, a precisão é calculada como a razão entre o número de N-gramas do texto gerado e o número total de N-gramas na referência original.

É importante ressaltar que, ao utilizar o quadrigramas para obter o resultado de *BLEU Score*, é possível obter um resultado mais preciso, visto que ele conseguirá captar com maior complexidade o contexto do resultado por comparar com uma sentença maior.

A fórmula de *BLEU Score* se dá no seguinte formato:

$$BP \cdot exp(\sum_{n=1}^{N} w_n log \cdot p_n) \tag{3.1}$$

Onde o valor de $p_n$ é o produtório dos N-gramas e o $w_n$ é o valor do N-grama escolhido

### 3.8.2 *ROUGE*

O *ROUGE*, cujo acrônimo em inglês é *Recall-Oriented Understudy for Gisting Evaluation* é uma métrica usada para avaliar a qualidade de resposta gerada por modelos de linguagens. Foi criada com propósito de avaliar textos de resumos e traduções, mas, além disso, é muito utilizada em funções mais amplas de PLN [22].

Essa métrica tem um valor entre 0 e 1 e quanto maior o resultado, maior é a correspondência entre a resposta gerada e a referência original. Ele também utiliza o cálculo de sobreposição de N-gramas entre o resultado gerado e o texto original. O *ROUGE* possui 3 variações principais, *ROUGE-N* que mede a sobreposição de N-gramas de *tokens*, *ROUGE-S* que mede a sobreposição de N-gramas que pode possuir até 2 espaços entre os *tokens* e *ROUGE-L* que mede a sequência de palavras mais longa. Nesse projeto será utilizado a métrica *ROUGE-L*, pois no lugar de exigir que as palavras estejam conectados sucessivamente como acontece em algumas técnicas de comparação de texto, ela foca em encontrar correspondências de palavras em sequência, permitindo lidar com variações na ordem das palavras, a fim de avaliar a coesão dos textos gerados com mais eficiência.

A fórmula de *ROUGE-L* situado na Equação 3.2 se dá no seguinte formato:

$$\text{ROUGE-L} = \frac{(1+\beta^2) R_{LCS} P_{LCS}}{R_{LCS} + \beta^2 P_{LCS}} \tag{3.2}$$

Em que, $LCS(reference, hypothesis)$ representa a sequência de palavras mais longa comparando o texto gerado pelo LLM com o referência original. Já, os $P_{LCS}$ e $R_{LCS}$ localizados nas Equações 3.3 e 3.4 indicam precisão e *recall* respectivamente. Por fim, o parâmetro $\beta$ é utilizado para determinar o peso do *recall* ou precisão e quando esse valor for 1, tem-se pesos iguais para ambos os fatores, resultando em média harmônica do *ROUGE-L*, equivalente ao seu *F1-Score* [9] ao qual será utilizado para avaliação ao longo do projeto.

$$P_{LCS} = \frac{LCS(reference, hypothesis)}{l^{unigram}_{hypothesis}} \tag{3.3}$$

$$R_{LCS} = \frac{LCS(reference, hypothesis)}{l^{unigram}_{reference}} \tag{3.4}$$

# 4

# Metodologia e Desenvolvimento

Nesse capítulo serão discutidos sobre a metodologia utilizada ao longo do projeto, incluindo as explicações sobre os passos utilizados e o motivo da escolha de cada um desses processos. Serão explorados, então, as principais etapas demonstradas na Figura 4.1 que vão desde a extração e limpeza dos dados até a geração das respostas dos LLM.

## 4.1 Extração e Processamento dos Dados

A partir do *github* contido no artigo *Leaflet of Medicine Corpus* [34], foram obtidos as informações sobre 1050 diferentes medicamentos, onde cada um deles está armazenado em um arquivo de texto separado. No entanto, foi necessário aplicar as técnicas de *scraping* explicados na Seção 3.1.1 visto que os documentos continham informações que não eram relevantes para o trabalho. Desse modo, foi obtido de forma automatizada apenas as informações importantes de cada arquivo de texto no qual as informações contêm dados como nome do remédio, uma breve definição, contraindicações, posologia, efeitos colaterais, formas de armazenamento e a sua composição. Portanto, a fim de facilitar o processamento de dados, foi optado em salvar as informações citadas anteriormente em um arquivo de formato Comma-Separated Values (CSV), em que foi estruturado com as seguintes colunas: "*Name*", "*Definition*", "*Contraindication*", "*Posology*", "*Side Effects*", "*Storage*", "*Composition*", correspondendo a cada uma das informações coletadas. Cada uma das linhas contém informações específicas sobre um determinado medicamento, criando assim uma base de dados estruturada que servirá para o restante do projeto.

Depois de obter as informações necessárias, foram aplicadas algumas técnicas para preparar os dados de forma que possam ser utilizados a fim de trazer o melhor resultado posteriormente. Dessa forma, foi feito a remoção de *tags* HTML, para que as respostas geradas pelos LLM consistissem apenas em texto limpo, livre desses elementos indesejados. Além disso, foi empregada a técnica de *chunking* explicada na Seção 3.2.1, para que pudesse diminuir os textos, fazendo com que os LLM conseguissem processar as informações, sem resultar na perda de informações importantes e melhorando na eficiência do processamento.

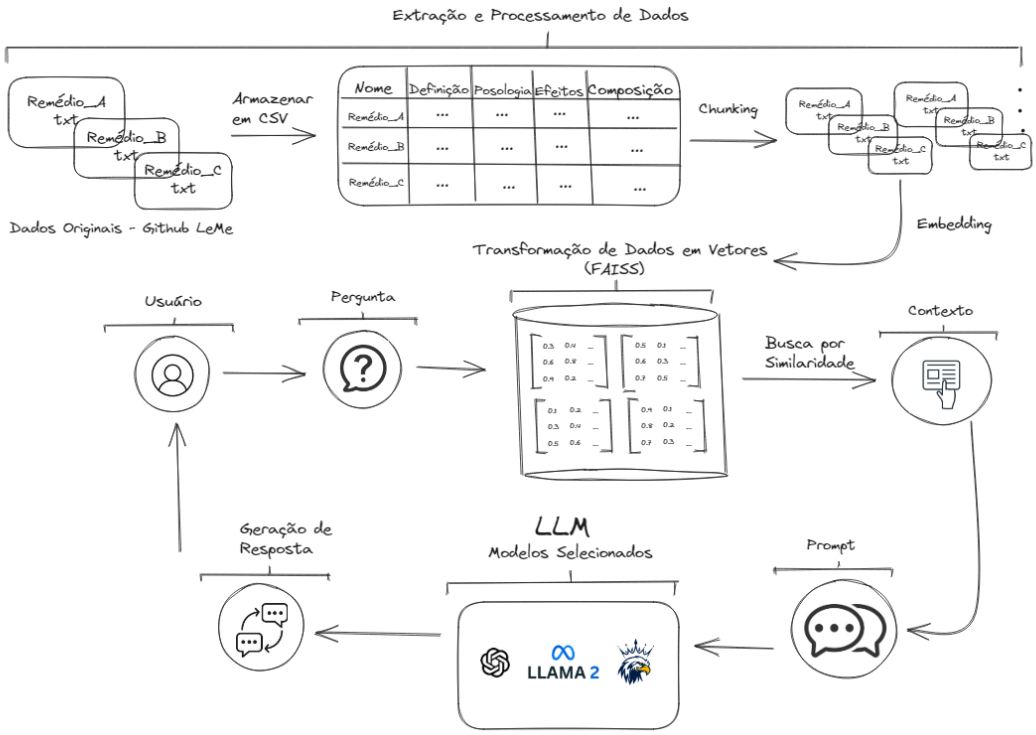


**Figura 4.1:** Fluxograma do Desenvolvimento do Projeto

## 4.2 Transformação de Dados em Vetores

Em seguida, foram escolhidos os modelos para aplicar na etapa de *embedding*, como mostrado na Seção 3.2.2, que transforma os dados dos remédios em representações vetoriais, permitindo que os LLM consigam "entender"o contexto, capturando as relações semânticas presente nas informações contidas no arquivo *Comma-Separated Values* (CSV). Os algoritmos de *embedding* que foram escolhidos durante o projeto foram o de *"text-embedding-ada-002"* e o de *"multi-qa-mpnet-base-dot-v1"*, pois são aqueles que conseguem gerar resultados adequados na língua portuguesa visto que eles foram treinados com base de dados de multilinguagem.

Ao transformar os *tokens* em formato vetorial, eles foram indexados em um banco de dados de vetores, explicado na Seção 3.3, utilizando a biblioteca *Facebook AI Similarity Search* (FAISS) [16] que foi desenvolvida pela equipe de *Facebook*, o qual consegue realizar uma busca por similaridade de forma eficiente ao permitir um armazenamento adequado e uma recuperação rápida dos vetores. Ao aplicar o método de *.similarity_search()* e passar para ele um parâmetro de entrada, o FAISS consegue realizar uma busca engenhosa utilizando os valores dos vetores previamente indexados que são os resultados do *chunking* e do *embedding*, no qual a quantidade de resultados retornados é definida pelo parâmetro K (k-vizinhos) [5].

Por fim, o FAISS utiliza a distância euclidiana cuja fórmula é mostrada na Equação 4.1 para calcular a similaridade entre a consulta (entrada) e todos os vetores indexados [6],

fornecendo assim os k-vizinhos com os menores valores de distâncias. Isso significa que há uma maior semelhança de texto entre a pergunta e as respostas geradas correspondentes, no qual o resultado é uma lista ordenada de respostas que mais se aparenta à consulta realizada, baseado nas concepções semânticas dos vetores. Uma vez geradas essas respostas, elas serão utilizadas para construir contextos na etapa de desenvolvimento de *prompt*.

$$dist(X_k, Y_k) = \sqrt{\sum_{k=1}^{n} (X_k - Y_k)^2} \tag{4.1}$$

## 4.3 Desenvolvimento de Prompt

Um dos aspectos cruciais ao interagir LLM para obter resultados na área específica é a criação de *prompts* de forma que possa guiar o modelo a performar de forma mais precisa [19]. Esse processo envolve a criação de *templates*, que são instruções específicas de como o modelo deve se comportar para realizar uma tarefa, sendo responsáveis, nesse projeto, pela atividade de responder as perguntas dos usuários na área de remédios.

Para ter um resultado mais adequado, é necessário definir um contexto para que o modelo tenha dados suficientes a fim de realizar uma consulta e responder às perguntas do usuário. Dessa forma, foi utilizado o primeiro resultado gerado pelo método *similarity_search()* da biblioteca FAISS [16] mostrado na Seção 4.2, produzindo informações que são condizentes com a pergunta feita. Juntamente com o contexto, o próprio *template* também irá conter a própria pergunta feita pelo usuário, possibilitando que o modelo possa incorporar a pergunta em sua resposta, trazendo resultados personalizados e relevantes.

```
Você é o meu assistente de IA que irá me fornecer as informações de forma resumida
sobre remédio que o usuário irá pergunta de forma educada e corente, dessa forma,
o usuário poderá pedir qualquer informação nessa lista ["Name", "Definition", "Con-
traindication", "Posology", "Side Effects", "Storage", "Composition"] ou utilizando
palavras similares
Sendo assim, forneça a resposta sobre a pergunta abaixo em português como se você
fosse um especialista da área:
Logo, baseado no contexto abaixo, responda a pergunta do usuário:
Contexto:
{context}
Pergunta:
{input}
Resposta:
```

Dessa forma, foram criadas 50 perguntas para abranger uma variedade de aspectos relacionados ao banco de dados gerado na Seção 4.1 associados a medicamentos. Para cada uma dessas perguntas, foi gerado um contexto correspondente pela biblioteca FAISS, sendo empregado tanto o modelo de *text-embedding-ada-002* quanto o modelo *multi-qa-mpnet-base-dot-v1* durante o processo de *embedding*. Essas perguntas e contextos foram salvos em arquivos de formato CSV como mostrado nas Tabelas 4.1 e 4.2, os quais, posteriormente, serão utilizados na etapa de geração de resposta que será mostrado na Seção 4.5.

**Tabela 4.1:** Exemplo de Base de Dados com Perguntas e Contexto usando *embedding* do *OpenAI*

| **Question** | **Similarity Search** |
|---|---|
| Quais são os efeitos colaterais do Zytiga? | Como todos os medicamentos, este medicamento pode... |
| Quais são os efeitos negativos ao tomar Saridon? | Foram ocasionalmente observadas reações alérgicas... |

**Tabela 4.2:** Exemplo de Base de Dados com Perguntas e Contexto usando *embedding* do *HuggingFace*

| **Question** | **Similarity Search** |
|---|---|
| Quais são os efeitos colaterais do Zytiga? | pode levar a colapso com perigode vida). Os indivíduos que manipulem rotineiramente... |
| Quais são os efeitos negativos ao tomar Saridon? | eles são potencialmente graves – assim, se tiver qualquer destes sintomas... |

## 4.4 Modelos Selecionados

Foram usados diversos modelos durante a realização desse trabalho, possuindo então dois grupos principais: aqueles que são oferecidos pela plataforma *HuggingFace* mostrada na Seção 3.7.1 que é conhecida por seus modelos de código aberto, e o modelo *gpt-3.5-turbo*, disponibilizado pela empresa *OpenAI* como mostrada na Seção 3.5.2. Para os modelos de *huggingface* foram usados os modelos *falcon-7b*, *llama-2-7b-hf* e *llama-2-13b-hf*, devido às suas popularidades e eficácias comprovadas ao realizar tarefas de geração de resposta, como evidenciada na figura 4.2. O *ranking* desses modelos foi obtido no momento da realização do projeto, assim como com a qualidade dos resultados que normalmente são obtidos por esses algoritmos.

| T | Model | Average | ARC | HellaSwag | MMLU | TruthfulQA |
|---|---|---|---|---|---|---|
| | meta-llama/Llama-2-7b-hf | 53.4 | 53.07 | 77.74 | 43.8 | 38.98 |
| | golaxy/gowizardlm | 53.06 | 49.74 | 71.9 | 42.96 | 47.66 |
| | stabilityai/stablelm-base-alpha-7b-v2 | 51.5 | 47.35 | 77.08 | 45.1 | 36.46 |
| | YeungNLP/firefly-llama2-7b-pretrain | 50.89 | 48.63 | 74.83 | 41.04 | 39.08 |
| | llama-7b | 49.72 | 51.02 | 77.82 | 35.71 | 34.33 |
| | DevaMalla/llama-base-7b | 49.69 | 50.94 | 77.8 | 35.67 | 34.34 |
| | openlm-research/open_llama_7b_v2 | 48.18 | 43.69 | 72.2 | 41.29 | 35.54 |
| | baichuan-inc/Baichuan-7B | 47.38 | 48.7 | 69.02 | 43.59 | 36.23 |
| | mosaicml/mpt-7b | 47.38 | 47.7 | 77.57 | 30.8 | 33.44 |
| | tiiuae/falcon-7b | 47.01 | 47.87 | 78.13 | 27.79 | 34.26 |
| | openlm-research/open_llama_7b | 46.08 | 47.01 | 71.98 | 30.49 | 34.85 |
| | togethercomputer/RedPajama-INCITE-Base-7B-v0.1 | 44.65 | 46.25 | 71.63 | 27.68 | 33.03 |

**Figura 4.2:** *Leaderboard* dos Modelos 7B do HuggingFace

## 4.5 Geração de Resposta

Nessa etapa de geração de resposta, será aplicada todos os procedimentos realizados nos estágios anteriores com o auxílio da biblioteca *LangChain* como apresentado na Seção 3.7.2 que atua como um conector eficiente entre o contexto, o *prompt* criado, a pergunta feita pelo usuário e os modelos de LLM selecionados, como por exemplo: *falcon-7b*, *llama-2-7b-hf*, *llama-2-13b-hf* ou *gpt-3.5-turbo*.

Cada um dos modelos foram introduzidos os mesmos valores dos parâmetros, sendo eles: *temperature*: 0.1, *max_new_tokens*: 512, *repetition_penalty*: 1.2, *top_p*: 0.95. E, abaixo, será detalhado sobre o que representa cada um desses valores [14].

- *Temperature*: Controla a aleatoriedade dos resultados gerados pelo modelo. Quanto menor valor, representa que é menos provável que o modelo gere palavras inesperadas ou incomuns, mas possa aumentar números de palavras repetidas. Quanto maior valor, representa que o texto gerado pode ser mais criativo, levando a respostas mais incoesas.

- *Max_New_Tokens*: Controla o tamanho máximo das respostas geradas pelo modelo em termos de *tokens*. Ao limitar o número de *tokens*, faz com que os modelos evitem de gerar respostas muito longas.

- *Repetition_Penalty*: Controla a tendência do modelo de repetir palavras ou frases em suas respostas.

- *Top_p*: Controla a probabilidade de escolher as palavras mais prováveis em uma determinada etapa da geração de texto.

Para mostrar alguns dos resultados gerados, serão criadas duas Tabelas abaixo, Tabela 4.3 e Tabela 4.4, em que cada uma delas conterão uma pergunta que será aplicada para os

quatro modelos da Seção 4.4 e cada Tabela terá resultados obtidos através dos dois modelos de *embedding* como mostrado na Seção 4.3. Dessa forma, é possível a comparação e a avaliação das respostas geradas pelos modelos em diversos cenários, enriquecendo a análise e a compreensão de suas capacidades.

**Tabela 4.3:** Exemplo de Respostas Geradas pelos Modelos utilizando *embedding* do *OpenAI*

| **Embedding - Text-Embedding-Ada-002** | |
|---|---|
| **Pergunta** | **Quais são os efeitos colaterais do Zytiga?** |
| Resposta Falcon-7b | [’- "Níveis baixos de potássio no sangue"’, ’ - "Níveis elevados de gordura no sangue"’, - "Níveis elevados de gordura no sangue"’, ’ - "Níveis elevados de gordura... |
| Resposta LLaMA2-7b | O Zytiga possui efeitos colaterais comuns, mas não todos os pacientes os experimentam. Alguns dos efeitos colaterais mais comuns incluem: Níveis baixos de potássio no sangue,... |
| Resposta LLaMA2-13b | Os efeitos colaterais comuns associados ao uso de Zytiga incluem fraqueza muscular, espasmos musculares, desequilibrância, verti-gensentimentos de vontade, náuseas, diarréia,... |
| Resposta GPT-3.5-Turbo | ’Os efeitos colaterais do Zytiga incluem fraqueza nos músculos, espasmos nos músculos, ritmo irregular do coração (palpitações), fluidos nas pernas ou pés, níveis baixos de potássio no sangue,... |

**Tabela 4.4:** Exemplo de Respostas Geradas pelos Modelos utilizando *embedding* do *HuggingFace*

| **Embedding - Multi-QA-Mpnet-Base-Dot-V1** | |
|---|---|
| **Pergunta** | **Quais são os efeitos colaterais do Zytiga?** |
| Resposta Falcon-7b | ’- Zytiga é um medicamento que é utilizado para o tratamento do benigno e não-benigno prostatitis. - O efeito colaterário mais comum é a náusea, mas também pode estar associado com a perda de apetite, a dor de estômago... |
| Resposta LLaMA2-7b | O Zytiga pode causar: - Náuseas; - Vômitos; - Dor de cabeça; - Dores de cabeça; - Fadiga; - Febre; - Tontura;... |
| Resposta LLaMA2-13b | 1) Efeitos adversos graves; 2) Efeitos adversos moderados; 3) Efeitos adversos leves; 4) Outros efeitos adversos; ”’... |
| Resposta GPT-3.5-Turbo | ’ Os possíveis efeitos colaterais do Zytiga incluem colapso com perigo de vida. Indivíduos que manipulam rotineiramente as formulações em pó estão mais predispostos,... |

# 5

# Avaliação dos Modelos

A partir dos resultados obtidos nas Tabelas 4.3 e 4.4, geradas pelos modelos de LLM escolhidos na Seção 4.4, foi utilizado nessa etapa de desempenho as métricas de *BLEU Score* e *ROUGE-L*, sendo que nesta última métrica foi empregado o valor de *F1-Score*, como mostrado na Seção de 3.8, para ter ideia da performance desempenhado pelos modelos.

Sendo assim, foi criada uma tabela denominada de *ground_truth*, como mostrado na Tabela 5.1, em que foi selecionado uma pequena porção de dados originais diretamente da fonte, consistindo em 50 perguntas específicas relacionadas a um tópico específico. Essa tabela completa pode ser acessada no *Github* do projeto[1], como por exemplo efeitos colaterais de um determinado medicamento. O objetivo dessa tabela é poder comparar esses textos obtidos de fonte original com as respostas geradas pelos modelos para permitir a avaliação do quão precisa está a resposta obtida ao emparelhar com texto original.

Em seguida, a partir da Tabela de 5.1, foi realizado uma avaliação de desempenho sobre cada um dos modelos selecionados, gerando resultados de *BLEU Score* e *ROUGE-L Score* para essas 50 perguntas. Após comparar as respostas dos modelos com o texto original, foram obtidos os valores da média de cada um dos modelos de acordo com o algoritmo de *embedding* que foi aplicado. Dessa forma, a partir das Tabelas 5.2 e 5.3, é possível comparar a performance de cada um dos modelos que foram utilizados durante o projeto.

[1] Disponível em https://github.com/LinJTF/MedicinesLLM/blob/main/data/ground_truth_medicines.csv

**Tabela 5.1:** Exemplo da Tabela de *Ground_Truth*

| ***Question*** | ***Original Info*** |
|---|---|
| Quais são os efeitos colaterais do Zytiga? | Como todos os medicamentos, este medicamento pode causar efeitos secundários, no entanto estes nãose manifestam em todas as pessoas. Pare de tinar ZYTIGA e dirija-se imediatamente... |
| Quais são os efeitos negativos ao tomar Saridon? | Foram ocasionalmente observadas reações alérgicas da pele (exantema, urticária). Ocorreram, muito raramente, problemas respiratórios ou reações do sistema circulatório em indivíduos... |
| Em que situações o remédio de Isotretinoína é contraindicado? | Não tome Isotretinoína Aurovitas Se está grávida ou a amamentar se estiver apta a engravidar, deve seguir as precauções do "Programa de gravideze prevenção", ver a secção "Advertências e precauções".... |
| Qual é a dose recomendada de Citalopram Genedec? | A dose recomendada é: Adultos Tratamento da depressão A dose habitual é de 20 mg por dia. O seu médico pode aumentar esta dose até um máximo de 40 mg por dia.... |

**Tabela 5.2:** Resultado dos Modelos Utilizando *Embedding* do *OpenAI*

| ***Embedding - Text-Embedding-Ada-002*** | | |
|---|---|---|
| **Modelos** | ***BLEU SCORE*** | ***ROUGE-L SCORE*** |
| **Falcon-7b** | 0.145494 | 0.196746 |
| **LLaMA2-7b** | 0.000849 | 0.174053 |
| **LLaMA2-13b** | 0.000851 | 0.137555 |
| **GPT-3.5-Turbo** | 0.213611 | 0.224693 |

**Tabela 5.3:** Resultado dos Modelos Utilizando *Embedding* do *HuggingFace*

| ***Embedding - Multi-QA-Mpnet-Base-Dot-V1*** | | |
|---|---|---|
| **Modelos** | ***BLEU SCORE*** | ***ROUGE-L SCORE*** |
| **Falcon-7b** | 0.093141 | 0.109865 |
| **LLaMA2-7b** | 0.001786 | 0.113307 |
| **LLaMA2-13b** | 0.000827 | 0.080995 |
| **GPT-3.5-Turbo** | 0.000832 | 0.138178 |

Logo, utilizando esses resultados obtidos nas Tabelas 5.2 e 5.3, é possível avaliar quão próximo os resultados gerados se assemelham com os textos de referência a partir da Tabela 5.1. Essa análise se torna crucial para que possa compreender a qualidade das respostas geradas, a fim de determinar quais modelos e *embedding* são os mais adequados para a tarefa específica em questão, no caso desse projeto, a tarefa de responder às perguntas dos usuários sobre um banco de medicamentos.

Dessa forma, observando os resultados através das Tabelas 5.2 e 5.3, é possível notar que a utilização do *embedding* do *OpenAI* foi superior ao desempenho obtido com o *embedding* do *Multi-QA-Mpnet-Base-Dot-V1*. Essa diferença de performance foi evidenciada em várias perguntas, em que o contexto gerado pelo *embedding* do *OpenAI* resultou em respostas mais precisas e coerentes, refletindo nas pontuações do *BLEU Score* e *Rouge-L* mais elevadas. Uma das razões que levou o *embedding* do *Multi-QA* ter tido uma performance inferior ao aplicado em LLM quando comparado com resultado do *embedding* do *OpenAI* pode ser devido a diferença nas dimensões desses *embeddings*, enquanto que *Multi-QA-Mpnet-Base-Dot-V1* tem 768 dimensões, o *embedding* de *OpenAI* possui 1536 dimensões [27]. O termo "dimensão"se refere ao tamanho do vetor que possa armazenar esses valores numéricos [33, 23]. Logo, em geral, os modelos de *embeddings* que possuem dimensões maiores têm mais capacidade de representação e podem capturar contextos mais complexos, mas exigem também mais recursos computacionais [23, 27]. Por outro lado, *embeddings* com dimensões menores são capazes de serem mais eficientes em termos de recursos, mas podem trazer resultados inferiores.

Uma observação que pode ser feita ao verificar o desempenho dos modelos a partir das métricas de *BLEU Score* e *ROUGE-L*, como pode ser visto nas Figuras 5.1 e 5.2 é a diferença nas suas pontuações. Em algumas perguntas específicas, é possível notar que o *BLEU Score* pode demonstrar valores inferiores, como por exemplo os modelos *LLaMA2-7b* e *LLaMA2-13b* ao comparar com o resultado do *ROUGE-L*, que se mantém relativamente estável em sua faixa de valor. Como explicado na Seção 3.8.1 e 3.8.2, essa diferença pode ser justificada pelo propósito das métricas. O *BLEU Score* avalia a sobreposição de palavras ou frases entre o resultado gerado pelo modelo e um conjunto de referência do texto original [2], a qual foi utilizada a Tabela 5.1 no projeto em questão, em outras palavras, essa métrica mede a precisão, levando em conta a correspondência das palavras nas respostas geradas pelo modelo com as palavras nos textos de referência fornecido. Enquanto isso, a métrica de *ROUGE-L* [9] mede a correspondência das palavras nas referências originais e compara com as palavras nas respostas geradas pelo modelo, aplicando o conceito de considerar a subsequência mais longa em comum (*Longest Common Subsequence* (LCS)) entre a resposta gerada e o texto de referência, tornando possível realizar a avaliação preservando a semântica da frase mesmo que a ordem das palavras sejam variadas.

Adicionalmente, a partir das Tabelas 5.2 e 5.3, foram criadas também os gráficos de *boxplots*. Esses gráficos foram apresentados em duas Figuras distintas, 5.1 e 5.2, que permitem a visualização dos desempenhos dos modelos em pares por conta dos dois *embeddings* utilizados nos modelos ao longo do projeto, sendo a cor azul o modelo aplicando o *embedding* de *OpenAI* e a cor vermelha o modelo aplicando o *embedding* de *Multi-QA-Mpnet-Base-Dot-V1*. Através desses gráficos, foi possível avaliar o desempenho com base nos indicadores de *BLEU Score* e *ROUGE Score*, uma visão geral da eficácia dos modelos sob diferentes contextos de texto. Logo, essa análise permite o entendimento da importância da escolha de *embedding* na obtenção de

respostas precisas e relevantes gerados pelos modelos de linguagem, com base nas necessidades dos usuários, mostrando a complexidade da otimização de LLM para aplicações de perguntas e respostas.

Para ilustrar ainda mais a questão da diferença dos desempenhos, a partir das Figuras 5.1 e 5.2, dentre os modelos *open-sources* utilizados, o modelo *falcon-7b* foi aquele que teve o melhor desempenho. Dessa forma, foram criadas as Tabelas 5.4 e 5.5 que contêm os resultados gerados, juntamente com o melhor resultado obtido através do método *.similarity_search()* e o texto original obtido através da tabela de *ground_truth*, como mostrada na Tabela 5.1. Com base nas amostras dessas Tabelas, é possível observar que mesmo que os resultados do *embedding* de *OpenAI* tenham sido melhores, existem algumas exceções que demonstram o contrário, em que o contexto obtido pelo *embedding* de *Multi-QA-Mpnet-Base-Dot-V1* gerou resultados superiores.

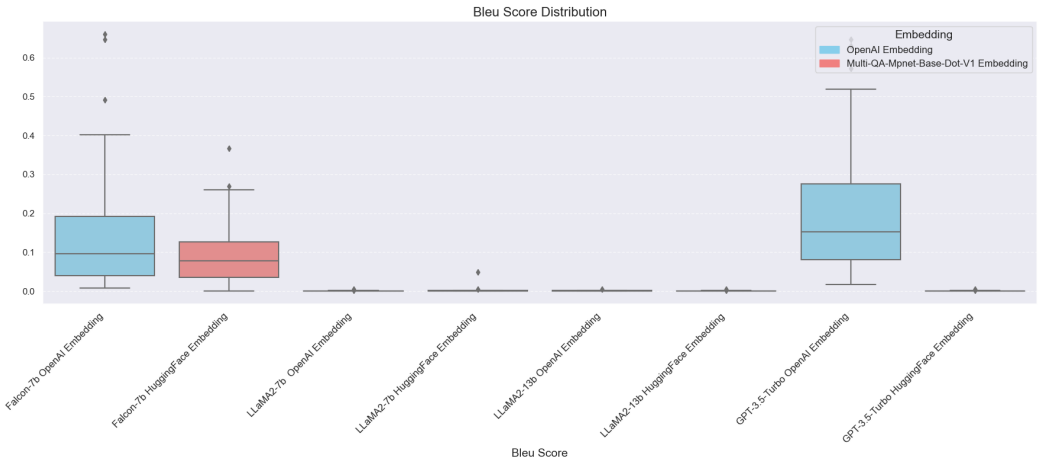


**Figura 5.1:** *Boxplot* do *BLEU Score* de Todos Modelos

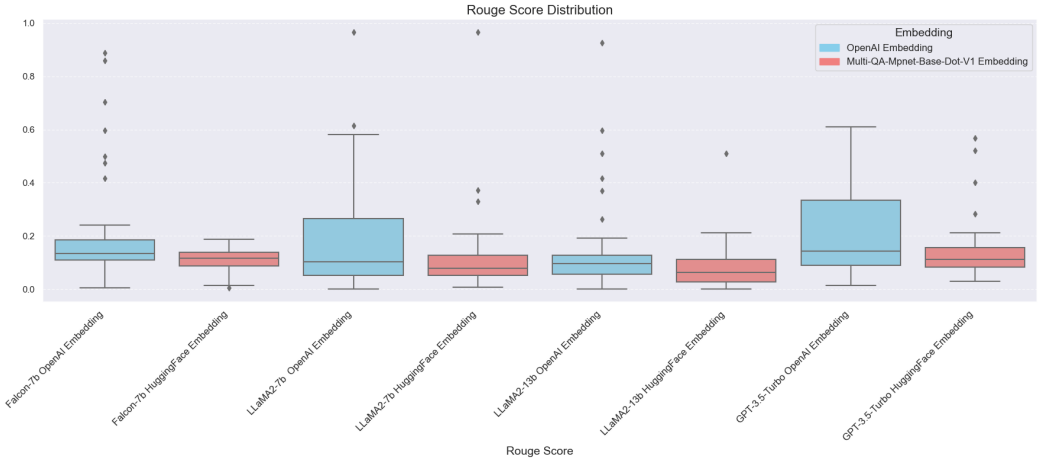


**Figura 5.2:** *Boxplot* do *Rouge-L Score* de Todos Modelos

**Tabela 5.4:** Exemplo de Melhor Resultado Usando *Embedding* do *OpenAI* no Modelo *Falcon-7b*

| ***Embedding - Text-Embedding-Ada-002*** | | | |
|---|---|---|---|
| **Pergunta** | **Resposta** | **Contexto K=1** | ***Ground Truth*** |
| **Quais são os efeitos colaterais do Zytiga?** | [’- "Níveis baixos de potássio no sangue"’, ’ - "Níveis elevados de gordura no sangue"’, ’ - "Níveis elevados de gordura no sangue"’, ’ - "Níveis elevados de gordura no sangue"’, ’ - "Níveis elevados de gordura no sangue"’,... | Como todos os medicamentos, este medicamento pode causar efeitos secundários, no entanto estes não se manifestam em todas as pessoas. Pare de tomar ZYTIGA e dirija-se imediatamente a um médico caso verifique: Fraqueza nos músculos,... | Como todos os medicamentos, este medicamento pode causar efeitos secundários, no entanto estes nãose manifestam em todas as pessoas.Pare de tomar ZYTIGA e dirija-se imediatamente a um médico caso verifique: Fraqueza nos músculos,... |

**Tabela 5.5:** Exemplo de Melhor Resultado Usando *Embedding* do *Multi-QA* no Modelo *Falcon-7b*

| ***Embedding - Multi-QA-Mpnet-Base-Dot-V1*** | | | |
|---|---|---|---|
| **Pergunta** | **Resposta** | **Contexto K=1** | ***Ground Truth*** |
| **Quais são os efeitos negativos ao tomar Saridon?** | - Saridon é um medicamento que contém aula, que é um medicamento que contém aula, que é um medicamento que contém aula, que é um medicamento que contém aula, ... | eles são potencialmente graves – assim, se tiver qualquer destes sintomas: Contacte o seu médico o mais rapidamente possível. Efeitos secundários muito frequentes: (podem afetar mais de 1 em cada 10 pessoas)• Hemorragias nasais,... | Foram ocasionalmente observadas reações alérgicas da pele (exantema, urticária). Ocorreram, muito raramente, problemas respiratórios ou reações do sistemacirculatório em indivíduos sensíveis |

# 6

# Conclusão

Durante o projeto, foi realizado uma série de etapas cruciais para desenvolver uma aplicação que gerasse respostas mais precisas e confiáveis sobre informações relacionadas a medicamentos. Tais processos incluíram a etapa de obtenção de dados através das técnicas de *scraping* 3.1.1, a realização de *embedding* 3.2.2, *chunking* 3.2.1 e o armazenamento dessas representações numéricas em um banco de dados de vetores 4.2 utilizando a biblioteca FAISS. Além disso, foi selecionado diversos modelos de LLM para obter respostas que pudessem atender às necessidades dos usuários 4.5.

Através do Capítulo 5, foram utilizados as métricas de qualidade de linguagem, como *BLEU Score* e *Rouge Score* que forneceram as pontuações sobre a qualidade das respostas geradas pelos modelos. Desse modo, foi visto que, em geral, o LLM que performou melhor junto com o seu respectivo *embedding* (modelo *text-embedding-ada-002*) foi o de *OpenAI*, elaborando textos mais coerentes e relevantes.

Entretanto, dos modelos de código aberto, o modelo *falcon-7b* teve também o seu resultado expressivo. Em algumas situações, os resultados gerados pelo mesmo superou até os modelos do *OpenAI*. Isso mostra a eficiência dos modelos de código aberto disponíveis na comunidade de PLN, a fim de atender diferentes cenários de aplicação, como em situações em que o acesso a modelos de código fechado é limitado.

Logo, é relevante ressaltar que o projeto foi empregado a etapa de inferência de modelos para obter resultados. Para o futuro, é possível realizar uma otimização nas respostas geradas através da técnica de ajuste-fino, permitindo ajustar os modelos para tarefas específicas e melhorar ainda mais a precisão das respostas. Juntamente a isso, é possível também realizar uma busca pelos melhores valores dos hiperparâmetros usando técnicas como o *AutoML* [35] para aperfeiçoar a eficiência da aplicação, garantindo que os textos gerados pelos LLM forneçam informações confiáveis e relevantes.

Por fim, esse trabalho teve o propósito de fornecer respostas valiosas aos usuários, com potencial de mitigar o problema relacionado à questão de automedicação, promovendo práticas de saúde mais responsáveis e conscientes. Dessa forma, isso pode representar um avanço tanto

na área de medicina, quanto na promoção do bem-estar geral da sociedade.

# Referências

[1] A. Andoni, P. Indyk, and I. P. Razenshteyn. Approximate nearest neighbor search in high dimensions. *CoRR*, abs/1806.09823, 2018.

[2] K. Blagec, G. Dorffner, M. Moradi, S. Ott, and M. Samwald. A global analysis of metrics used for measuring performance in natural language processing, 2022.

[3] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners, 2020.

[4] J. Camacho-Collados and M. T. Pilehvar. On the role of text preprocessing in neural network architectures: An evaluation study on text categorization and sentiment analysis, 2018.

[5] C. Coleman, E. Chou, J. Katz-Samuels, S. Culatana, P. Bailis, A. C. Berg, R. Nowak, R. Sumbaly, M. Zaharia, and I. Z. Yalniz. Similarity search for efficient active learning and search of rare concepts, 2021.

[6] D. Danopoulos, C. Kachris, and D. Soudris. *Approximate Similarity Search with FAISS Framework Using FPGAs on the Cloud*, pages 373–386. 08 2019.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[8] H. C. e Michele Aguiar e Reinaldo Geraldo e Cícero Freitas e Luciane Alcoforado e Dilvani Santos e Camila Barbosa e Clara Fonseca e Clarissa Aló e Erica Rangel e Ingrid Toledo e Marcela Feitosa e Carlos Rodrigues e Teresa Santos e Lúcio Cabral. AutomedicaÇÃo: Entendemos o risco? *Infarma - Ciências Farmacêuticas*, 18(9/10):17–20, 2013.

[9] M. Evtikhiev, E. Bogomolov, Y. Sokolov, and T. Bryksin. Out of the BLEU: How should we assess quality of the code generation models? *Journal of Systems and Software*, 203:111741, sep 2023.

[10] J. Gao and C. Long. High-dimensional approximate nearest neighbor search: with reliable and efficient distance comparison operations, 2023.

[11] T. George. What are the limitations of chatgpt? *Scribbr*, Aug 2023.

[12] Y. Goldberg. *Neural Network Methods for Natural Language Processing*, volume 37 of *Synthesis Lectures on Human Language Technologies*. Morgan & Claypool, San Rafael, CA, 2017.

[13] A. Haleem, M. Javaid, M. Asim Qadri, R. Pratap Singh, and R. Suman. Artificial intelligence (ai) applications for marketing: A literature-based study. *International Journal of Intelligent Networks*, 3:119–132, 2022.

[14] hugging_face. Detailed parameters — huggingface.co. `https://huggingface.co/docs/api-inference/detailed_parameters`. [Accessed 15-09-2023].

[15] Z. Jiang, F. F. Xu, L. Gao, Z. Sun, Q. Liu, J. Dwivedi-Yu, Y. Yang, J. Callan, and G. Neubig. Active retrieval augmented generation, 2023.

[16] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with gpus, 2017.

[17] J. Kaddour, J. Harris, M. Mozes, H. Bradley, R. Raileanu, and R. McHardy. Challenges and applications of large language models, 2023.

[18] M. C. Keiper. Chatgpt in practice: Increasing event planning efficiency through artificial intelligence. *Journal of Hospitality, Leisure, Sport  Tourism Education*, 33:100454, 2023.

[19] C. S. Krishna. Prompt generate train (pgt): Few-shot domain adaption of retrieval augmented generation models for open book question-answering, 2023.

[20] E. Leonardi. Aproximadamente 90% dos brasileiros realizam automedicação, 2022.

[21] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[22] C.-Y. Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[23] C. Liu, S. Yu, Y. Huang, and Z.-K. Zhang. Effective model integration algorithm for improving link and sign prediction in complex networks. *IEEE Transactions on Network Science and Engineering*, PP:1–1, 07 2021.

[24] C. Lotfi, S. Srinivasan, M. Ertz, and I. Latrous. *Web Scraping Techniques and Applications: A Literature Review*, pages 381–394. 01 2021.

[25] S. S. Manathunga and Y. A. Illangasekara. Retrieval augmented generation and representative vector summarization for large unstructured textual data in medical education, 2023.

[26] A. Mishra, J. Cha, and S. Kim. Single neuron for solving xor like nonlinear problems. *Computational Intelligence and Neuroscience*, 2022, 04 2022.

[27] N. Muennighoff, N. Tazi, L. Magne, and N. Reimers. Mteb: Massive text embedding benchmark, 2023.

[28] K. Papineni, S. Roukos, T. Ward, and W. J. Zhu. Bleu: a method for automatic evaluation of machine translation. 10 2002.

[29] G. Penedo, Q. Malartic, D. Hesslow, R. Cojocaru, A. Cappelli, H. Alobeidli, B. Pannier, E. Almazrouei, and J. Launay. The refinedweb dataset for falcon llm: Outperforming curated corpora with web data, and web data only, 2023.

[30] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. 2018.

[31] L. A. Ramshaw and M. P. Marcus. Text chunking using transformation-based learning, 1995.

[32] K. Safjan. Understanding retrieval-augmented generation (rag) empowering llms. *Krystian's Safjan Blog*, 2023.

[33] E. Sezerer and S. Tekir. A survey on neural word embeddings, 2021.

[34] A. SimÕes and P. Gamallo. LeMe-PT: A Medical Package Leaflet Corpus for Portuguese. In R. Queirós, M. Pinto, A. Simões, F. Portela, and M. J. a. Pereira, editors, *10th Symposium on Languages, Applications and Technologies (SLATE 2021)*, volume 94 of *Open Access Series in Informatics (OASIcs)*, pages 10:1–10:10, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[35] A. Tornede, D. Deng, T. Eimer, J. Giovanelli, A. Mohan, T. Ruhkopf, S. Segel, D. Theodorakopoulos, T. Tornede, H. Wachsmuth, and M. Lindauer. Automl in the age of large language models: Current challenges, future opportunities and risks, 2023.

[36] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi,

A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[37] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2023.

[38] H. Wang, J. Li, H. Wu, E. Hovy, and Y. Sun. Pre-trained language models and their applications. *Engineering*, 2022.

[39] J. Wang, X. Yi, R. Guo, H. Jin, P. Xu, S. Li, X. Wang, X. Guo, C. Li, X. Xu, K. Yu, Y. Yuan, Y. Zou, J. Long, Z. L. Yudong Cai, Z. Zhang, Y. Mo, J. Gu, R. Jiang, Y. Wei, and C. Xie. Milvus: A purpose-built vector data management system. *Purdue University*, 2021.

[40] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, and D. C. Schmidt. A prompt pattern catalog to enhance prompt engineering with chatgpt, 2023.

[41] J. Xie, P. Cheng, X. Liang, Y. Dai, and N. Du. Chunk, align, select: A simple long-sequence processing method for transformers, 2023.

[42] J. Ye, X. Chen, N. Xu, C. Zu, Z. Shao, S. Liu, Y. Cui, Z. Zhou, C. Gong, Y. Shen, J. Zhou, S. Chen, T. Gui, Q. Zhang, and X. Huang. A comprehensive capability analysis of gpt-3 and gpt-3.5 series models, 2023.

[43] L. Zhao, W. Alhoshan, A. Ferrari, and K. J. Letsholo. Classification of natural language processing techniques for requirements engineering, 2022.

[44] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. A comprehensive survey on transfer learning, 2020.