Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Utilização de RAG na criação de chatbot para auxílio na atenção básica de saúde

Marcelo Junqueira Ferreira, Vinicius Lima Passos

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof. Dr. Thiago de Paulo Faleiros

Brasília
2025

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Utilização de RAG na criação de chatbot para auxílio na atenção básica de saúde

Marcelo Junqueira Ferreira, Vinicius Lima Passos

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof. Dr. Thiago de Paulo Faleiros (Orientador)
CIC/UnB

Prof. Dr. Luis Paulo Faina Garcia
CIC/UnB

Dr. Edna Dias Canedo
CIC/UnB

Prof. Dr. Marcelo Grandi Mandelli
Coordenador do Bacharelado em Ciência da Computação

Brasília, 13 de Fevereiro de 2025

Dedicatória

Vinícius Lima

Dedico este trabalho a meus familiares, cujo apoio incondicional e incentivo foram essenciais em minha jornada acadêmica. Aos amigos que estiveram ao meu lado nos momentos desafiadores e aos professores que compartilharam conhecimento e inspiração ao longo do caminho.

Dedicatória Marcelo Junqueira

Dedico este trabalho aos meus pais, que viram diariamente meu esforço e sacrifício, buscando sempre evoluir e deixá-los orgulhosos.

Agradecimentos

Agradecimentos Vinícius Lima

Agradeço à minha família pelo suporte e incentivo contínuo, aos colegas e amigos pelo companheirismo, e aos professores que contribuíram para meu aprendizado. Também expresso minha gratidão a todos que, direta ou indiretamente, ajudaram na realização desta monografia.

Agradecimentos Marcelo Junqueira

Agradeço à minha família pelo incentivo e pelo suporte desde sempre.

Agradeço a todos os professores com quem tive a oportunidade de aprender e que moldaram quem eu sou hoje.

Agradeço aos meus colegas alunos da UnB e a meus colegas da CJR que me ajudaram a passar pelo curso de forma mais leve e tranquila.

Um agradecimento especial ao nosso professor orientador Thiago de Paulo Faleiros, que nos guiou nessa jornada.

Outro agradecimento muito especial à minha dupla Vinícius Lima, que me acompanhou desde o início do curso. Agradeço pela parceria, pela perseverança e pelo ótimo trabalho.

Resumo

Os sistemas de Recuperação Aumentada por Geração (RAG) têm se mostrado uma abordagem promissora para o desenvolvimento de chatbots que utilizam grandes modelos de linguagem (LLMs) como base para fornecer respostas contextualizadas. Este estudo investiga a viabilidade de um chatbot RAG aplicado ao contexto da saúde, avaliando seu desempenho técnico e explorando diferentes configurações de segmentação e indexação dos documentos de conhecimento. Para isso, foi desenvolvido um chatbot no Telegram que utiliza a um sistema de RAG para responder a perguntas, integrando o LLama3 e o VectorStore ChromaDB para recuperação eficiente da informação. O sistema foi avaliado por meio de uma pipeline automatizada utilizando o framework RAGAS, onde 18 configurações distintas do RAG foram testadas com 200 perguntas baseadas em documentos reais da área da saúde. Os resultados indicam que o chatbot apresentou um desempenho consistente e adequado para o suporte à saúde, sem diferenças estatisticamente significativas entre as diferentes configurações analisadas. No entanto, os desafios na avaliação automatizada de modelos de linguagem ainda representam um fator crítico, sugerindo a necessidade de investigações futuras que integrem avaliações humanas ao processo.

Palavras-chave: RAG, Avaliação de Modelos de Linguagem, RAGAS, Chatbots na Saúde, Atenção básica

Abstract

Retrieval-Augmented Generation (RAG) systems have proven to be a promising approach for developing chatbots that utilize large language models (LLMs) as a foundation to provide contextualized responses. This study investigates the feasibility of a RAG chatbot applied to the healthcare context, assessing its technical performance and exploring different configurations for segmenting and indexing knowledge documents. To this end, a Telegram chatbot was developed that employs a RAG system to answer questions, integrating LLama3 and the VectorStore ChromaDB for efficient information retrieval. The system was evaluated through an automated pipeline using the RAGAS framework, where 18 distinct RAG configurations were tested with 200 questions based on real healthcare documents. The results indicate that the chatbot demonstrated consistent and suitable performance for healthcare support, with no statistically significant differences among the various configurations analyzed. However, challenges in the automated evaluation of language models remain a critical factor, suggesting the need for future investigations that incorporate human assessments into the process.

Keywords: RAG, Healthcare Chatbots, Language Model Evaluation, RAGAS

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A aplicação de tecnologias avançadas no âmbito da saúde tem gerado transformações significativas na forma como os serviços são oferecidos e acessados. Em particular, o uso de LLMs apresenta um grande potencial para melhorar a eficiência e a qualidade dos cuidados de saúde. O Sistema Único de Saúde (SUS) no Brasil, com sua abrangência e complexidade, representa um cenário ideal para explorar essas tecnologias, visando enfrentar desafios históricos relacionados à integração, rastreamento e triagem no contexto da Atenção Primária à Saúde (APS).

1.1 Motivação

Aproximadamente 18,6 milhões de pessoas no Brasil apresentam algum tipo de deficiência, enfrentando significativas disparidades sociais e obstáculos no acesso aos serviços de saúde [IBGE,], nesse cenário a Rede de Cuidados à Pessoa com Deficiência (RCPD), implementada pelo Governo Federal em 2012, busca promover o acesso equânime e igualitário aos serviços de saúde, criando e articulando pontos de atenção. Contudo, passados mais de dez anos de sua criação, a RCPD ainda enfrenta dificuldades quanto à integração dos componentes da rede, definição de fluxos assistenciais e rastreamento eficiente das pessoas com deficiência ou risco [Mota and Bousquat, 2023].

Nesse contexto, surge a oportunidade de desenvolver ferramentas inovadoras que descentralizem o rastreamento e a triagem, integrem eficazmente os níveis de cuidado e utilizem tecnologias modernas para apoiar a Estratégia de Saúde da Família e, em especial, os Agentes Comunitários de Saúde (ACS). Este trabalho propõe o desenvolvimento de um chatbot baseado em RAG para apoiar o trabalho dos ACS, fornecendo informações contextuais e diretrizes com base em fontes confiáveis e regulamentações do Ministério da Saúde.

A aplicação de chatbots na área da saúde tem sido amplamente investigada na literatura, evidenciando tanto os benefícios quanto os desafios dessa abordagem. Estudos como [Gams et al., 2024] demonstram a viabilidade da integração de conhecimento médico em sistemas baseados em modelos generativos, destacando a utilização do RAG para aprimorar a precisão das respostas em aplicações clínicas. De forma semelhante, [Kim and Min, 2024] apresenta um sistema de RAG especializado na interpretação de regulamentações farmacêuticas, evidenciando o potencial da tecnologia na garantia da conformidade regulatória. Esses trabalhos demonstram diferentes provas de conceito para a aplicação de chatbots baseados em RAG em domínios específicos da saúde, reforçando a relevância da presente proposta para a atuação dos ACS no contexto brasileiro.

Para garantir a eficácia de um chatbot baseado em RAG, é fundamental realizar uma avaliação rigorosa do sistema. Contudo, avaliar sistemas RAG envolve a análise dos componentes de recuperação, geração e do sistema como um todo, esse processo de avaliação é multifacetado, exigindo uma consideração cuidadosa de diversas métricas e desafios específicos. Cada um desses aspectos representa dificuldades que complicam o desenvolvimento de um arcabouço abrangente de avaliação e *benchmarks* adequados.

Dentre as dificuldades encontradas na avaliação, é possível citar as dificuldades expostas em [Yu et al., 2024a], como a diversidade das fontes de informação, o que pode resultar na recuperação de conteúdos enganosos ou de baixa qualidade, tornando desafiadora a seleção e filtragem das informações mais pertinentes para o *retriever* e a verificação da fidelidade das respostas geradas pelo *generator* em relação ao contexto recuperados.

Além disso, há a necessidade de se avaliar a qualidade da resposta si, principalmente em domínios específicos que demandam a construção de bases de conhecimento direcionadas, como no contexto da saúde brasileira. A construção de tais bases requer um trabalho custoso de seleção de perguntas e trechos, enfrentando desafios significativos no acesso a informações de qualidade. Estudos como os elencados em [Lima et al., 2024] destacam a importância da integração e padronização de registros para superar essas dificuldades.

Dado que, conforme elencado anteriormente, há a necessidade de avaliação do sistema para garantir sua eficácia, foram elaboradas as seguintes *Research Questions*:

- RC1: Um chatbot baseado em RAG pode alcançar um bom desempenho em respostas a perguntas relacionadas à saúde, conforme métricas específicas?

  A confiabilidade das informações fornecidas por um chatbot desse tipo é crucial, uma vez que decisões relacionadas à saúde podem impactar diretamente a qualidade do atendimento prestado. Assim, responder a essa questão é essencial para validar a viabilidade do uso de um chatbot RAG nesse contexto.

- RC2: Entre as diferentes configurações testadas, é possível identificar uma configuração ideal que otimize a recuperação de informações e a geração de respostas?

  Como sistemas RAG combinam recuperação de informações e modelos generativos, a configuração desses componentes pode impactar significativamente o desempenho do chatbot. Fatores como o tamanho dos blocos de texto extraídos dos documentos, a métrica de similaridade utilizada na recuperação e o modelo de geração podem influenciar a qualidade das respostas.

  Assim, identificar a configuração ideal é essencial para garantir que o chatbot forneça respostas precisas, coerentes e contextualmente relevantes. Além disso, essa análise pode contribuir para o avanço da pesquisa na área, fornecendo diretrizes para o ajuste fino de sistemas RAG em domínios específicos da saúde.

1.2 Objetivo

Os objetivos principais desta monografia são explorar a viabilidade do uso de RAG na criação de um chatbot para apoio às atividades dos agentes comunitários de saúde, bem como avaliar seu desempenho, buscando as possíveis otimizações.

Foram elencados os seguintes objetivos específicos a serem concluídos ao longo da pesquisa:

- Desenvolver um chatbot baseado em RAG para responder perguntas médicas e auxiliar agentes comunitários de saúde no acesso rápido e confiável a informações.
- Disponibilizar a ferramenta em plataformas de mensagens instantâneas, como Telegram e WhatsApp, facilitando o acesso dos usuários.
- Geração automatizada de uma base testes para avaliação do sistema sem a necessidade de avaliação humana.
- Avaliar o desempenho do sistema desenvolvido por meio de métodos consolidados na área de avaliação de RAG e modelos de linguagem.

1.3 Estrutura do Documento

Este documento está organizado da seguinte forma:

- No capítulo 2, são apresentados os fundamentos teóricos relacionados à tecnologias de *Retrieval-Augmented Generation*, modelos de linguagem, .

- Os capítulos 3 e 4 descrevem a metodologia adotada, incluindo os processos de desenvolvimento do sistema, curadoria de dados e configuração experimental.
- No capítulo 5, são discutidos os resultados obtidos nos experimentos, com ênfase na análise de desempenho.
- Por fim, o capítulo 6 apresenta as conclusões do trabalho, ameças à validade do estudo e pequisa e sugestões para estudos futuros.

Capítulo 2

Referencial Teórico

Este capítulo apresenta os fundamentos teóricos necessários para a compreensão e desenvolvimento do *chatbot* proposto. Inicialmente, são abordados os conceitos de Processamento de Linguagem Natural, *large language models* e a arquitetura *transformer*, destacando os mecanismos que viabilizam sua eficiência na modelagem da linguagem. Em seguida, discute-se a abordagem RAG, detalhando seus principais componentes e seus métodos de avaliação, essenciais para medir a qualidade e a confiabilidade das respostas geradas.

2.1 Processamento de Linguagem Natural (PLN)

PLN é uma área da inteligência artificial que visa permitir que máquinas compreendam, interpretem e gerem linguagem humana. O campo combina conhecimentos de linguística, estatística e aprendizado de máquina para analisar e modelar a linguagem natural. Segundo [Jurafsky and Martin, 2025] o PLN abrange desde tarefas simples, como *tokenização* e análise sintática, até aplicações complexas, como tradução automática, *chatbots* e sumarização de texto.

2.1.1 Representação da Linguagem

A representação da linguagem é um aspecto fundamental do PLN, pois define como os textos são estruturados para que algoritmos possam processá-los e interpretá-los. Diferentes técnicas de representação foram desenvolvidas ao longo do tempo, desde abordagens baseadas em regras e estatísticas até representações vetoriais modernas utilizando aprendizado profundo. Nesta seção, algumas delas serão descritas.

Term Frequency-Inverse Document Frequency (TF-IDF).

Formas de representação de texto mais simples são feita baseadas em regras ou utilizando métodos estatísticos, dentre as quais é possível exemplificar o TF-IDF. A técnica TF-IDF é uma medida numérica usada para pontuar a importância de uma palavra em um documento com base na frequência com que ela apareceu naquele documento e em uma determinada coleção de documentos [Sammut and Webb, 2017]. Sendo definida pela seguinte fórmula:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$

Onde TF é a frequência de um termo t em um documento d, calculada como:

$$\text{TF}(t, d) = \frac{\text{Número de vezes que } t \text{ aparece em } d}{\text{Número total de termos em } d}$$

e IDF Mede a importância de um termo tt, calculada como:

$$\text{IDF}(t) = \log\left(\frac{N}{|\{d \in D : t \in d\}|}\right)$$

em que N é o número total de documentos no corpus e $|d \in D : t \in d|$ é o número de documentos onde o termo t aparece.

Word2Vec

A abordagem tradicional de representação vetorial como o TF-IDF não captura a semântica das palavras. Para resolver essa limitação, surgiram os *word embeddings*, representações densas que mapeiam palavras em um espaço vetorial contínuo, onde palavras semanticamente semelhantes estão próximas.

O Word2Vec utiliza representações vetoriais a partir do contexto da palavra em grandes corpora de texto [Mikolov et al., 2013]. Ele pode ser treinado de duas formas:

- **Continuous Bag of Words(CBOW)**: Prediz uma palavra com base em seu contexto.
- **Skip-gram**: Prediz palavras do contexto com base em uma palavra central.

Essas formas de treinamento utilizam a função de similaridade do produto escalar para aproximar palavras que ocorrem em contextos semelhantes, melhorando tarefas como tradução automática e análise semântica.

Representação baseada em modelos

Com o avanço das redes neurais profundas, surgiram representações contextuais mais sofisticadas, como as baseadas em *transformers*, das quais o BERT é um dos principais exemplos.

O BERT introduz uma abordagem bidirecional para a representação da linguagem, examinando o contexto em ambas as direções ao mesmo tempo [Devlin et al., 2019]. Esse treinamento permite que o modelo capture a relação semântica entre as palavras de uma frase, resultando em uma compreensão mais profunda do texto.

Utilizando apenas o *encoder* da arquitetura de *transformer*, cada palavra de uma sentença é transformada em um vetor denso de alta dimensionalidade através de um processo de *embedding*, que captura informações semânticas e posicionais do texto. Esse *embedding* é composto por três partes principais, apresentadas na Figura 2.1 e descritas a seguir:

- **Token Embedding**: Representa cada palavra no texto utilizando um vetor aprendido. O BERT usa o vocabulário *WordPiece*, onde palavras raras são divididas em sub-palavras, garantindo maior flexibilidade para lidar com palavras desconhecidas. Por exemplo, a palavra "desenvolvimento" pode ser dividida em ["des", "envolvimento"], e o modelo aprende representações para cada parte.
- **Segment Embedding**: Indica a qual sentença uma palavra pertence quando o modelo processa pares de sentenças. O modelo diferencia as frases adicionando um vetor específico para cada segmento.
- **Positional Embedding**: O BERT adiciona embeddings posicionais para indicar a ordem das palavras na sentença. Esses embeddings são vetores que codificam a posição de cada palavra na sequência, permitindo que o modelo capture informações sobre a ordem das palavras.

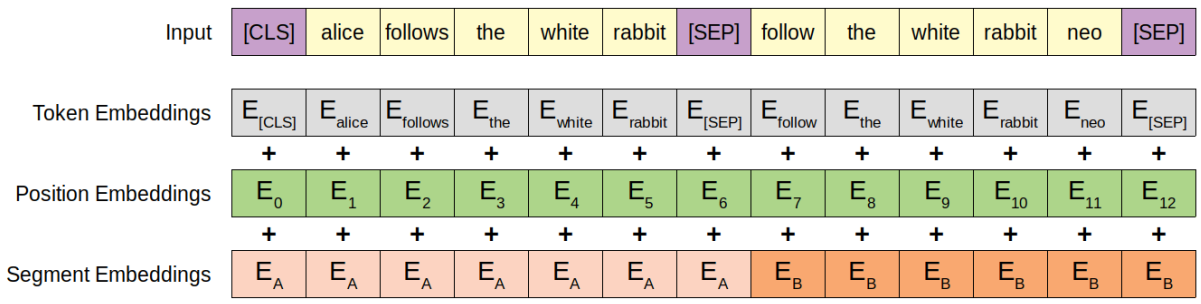

Figura 2.1: Exemplo de entrada do Bert

Em relação ao treinamento, ele ocorre previamente em grandes quantidades de dados textuais utilizando duas tarefas principais:

- **Masked Language Model (MLM)**: Para permitir que o modelo aprenda representações bidirecionais, 15% das palavras em uma frase são aleatoriamente mascaradas e o modelo deve prever a palavra original. Dessa forma, o BERT aprende a considerar todo o contexto de uma palavra, pois pode olhar para os dois lados da sentença.
- **Next Sentence Prediction (NSP)**: O modelo recebe pares de sentenças e deve determinar se a segunda sentença segue naturalmente a primeira. Esse treinamento auxilia em tarefas que exigem compreensão de múltiplas sentenças, como perguntas e respostas.

Devido à sua estrutura bidirecional e contextualizada, o BERT apresentou avanços significativos em diversas tarefas de PLN, incluindo análise de sentimentos, recuperação de informação, tradução automática e *chatbots*. Com o tempo, várias variações variações surgiram, como por exemplo o *RoBERTa*, em que há treinamento com maior quantidade de dados e mais longo, com a remoção do NSP [Zhuang et al., 2021].

A partir do modelo *RoBERTa*, foi desenvolvido o *XLM-RoBERTa*, o qual é uma versão multilingual do RoBERTa com desempenho comparável a modelos monolíngues [Conneau et al., 2020]. O *XLM-RoBERTa* foi utilizado para inicialização do modelo de embeddings utilizado neste estudo, o *intfloat/multilingual-e5-large*[1]. Esse *embedding*, apesar de ser multilingual, possui desempenho a par de outros modelos de estado da arte com foco apenas em inglês e tamanho similar [Wang et al., 2024].

2.2 Large language models (LLMs)

LLMs são sistemas avançados de inteligência artificial, projetados para compreender e gerar texto semelhante à linguagem humana. Esses modelos possuem uma quantidade massiva de parâmetros, na ordem de dezenas ou centenas de bilhões, assim obtendo um grande ganho de desempenho, além de "habilidades especiais" que não são percebidas em modelos menores [Zhao et al., 2024].

Esses modelos são construídos sobre arquiteturas de aprendizagem profunda, particularmente a arquitetura *Transformer* [Vaswani et al., 2017], que lhes permite capturar padrões intrincados na linguagem. Ao treinar em extensos conjuntos de dados de texto, os LLMs adquirem a capacidade de realizar uma ampla gama de tarefas de processamento de linguagem natural, incluindo geração de texto, tradução e resposta a perguntas.

[1] `https://huggingface.co/intfloat/multilingual-e5-large`

2.2.1 Transformers

A arquitetura *Transformer* revolucionou o campo de PLN, como descrito por [Vaswani et al., 2017] e permitiu o desenvolvimento de modelos como o GPT-4 [OpenAI et al., 2024] e o BERT [Devlin et al., 2019]. A sua principal inovação foi a substituição de modelos baseados em redes neurais recorrentes (RNNs) e convolucionais (CNNs) pelo mecanismo de *Self-Attention*, permitindo um treinamento mais eficiente, paralelização e captura de dependências de longo alcance em sequências textuais.

A arquitetura do *Transformer* é composta por dois blocos principais:

- **Encoder**: Extrai representações ricas do texto de entrada.
- **Decoder**: Utiliza essas representações para gerar sequências de saída, como em tarefas de tradução automática.

Como pode ser visto na Figura 2.2, ambos o *encoder* e o *decoder* são compostos pelas camadas *embedding*, *positional encoding*, *self atention* e *feed-Forward network*.

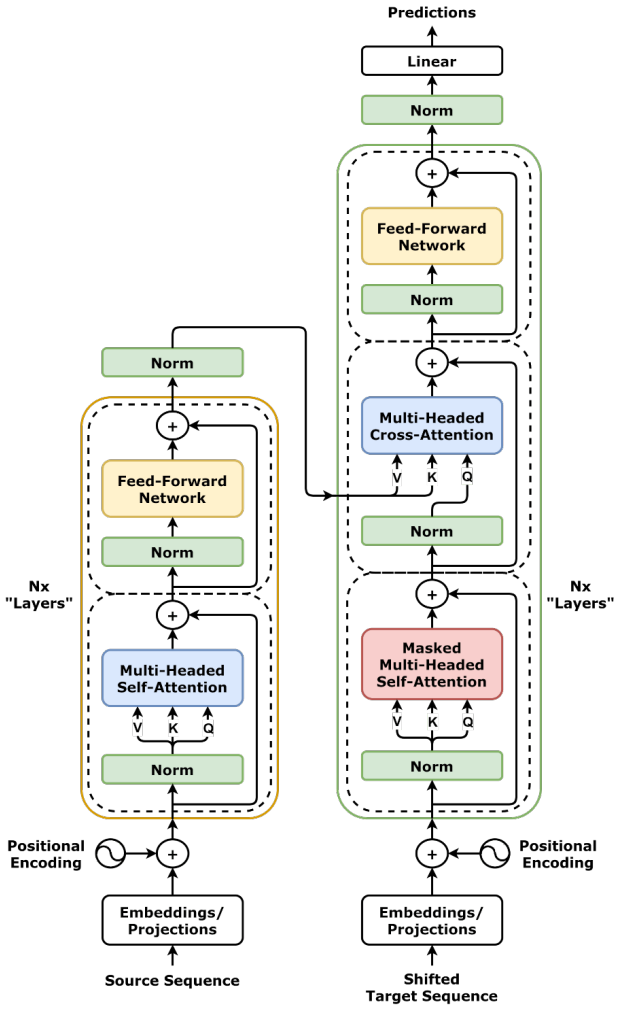

Figura 2.2: Arquitetura Transformer, imagem de [dvgodoy, 2025]

Embedding e Positional Encoding

Ao entrar no transformer, os tokens de entrada são convertidos em vetores numéricos através de uma camada de embeddig. Em seguida, para o modelo ter acesso a informações a respeito da ordem da sequência, a posição dos tokens precisa ser incorporada explicitamente.

Essa incorporação da posição é realizada pelo *Positional Encoding*, utilizando as fórmulas de variações seno e cosseno apresentadas a seguir, onde **pos** é a posição do token na sequência, **i** é o índice que identifica a dimensão atual dentro do vetor de embedding, **d** é o número total de dimensões do vetor de embedding e o **10000** é uma constante utilizada para definir a escala das funções seno e cosseno.

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d}}}\right)$$

Self-Attention e Multi-Head Attention

Os embeddings são, em seguida, submetidos a uma normalização. Na normalização a média e variância são calculados para cada camada ajustando os valores de ativação a fim de estabilizar o treinamento e melhorar a convergência do modelo [Ba et al., 2016], evitando o problema de explosão ou desaparecimento de gradientes. Os *embeddings* normalizados são então enviados para uma camada de *multi-headed self-atention*.

O mecanismo de atenção foi originalmente introduzido por [Bahdanau et al., 2016] com o objetivo de melhorar a tradução automática baseada em redes neurais, sendo posteriormente também utilizado no contexto da implementação da arquitetura *transformer*.

O mecanismo pode ser entendido como um filtro adaptativo, que foca seletivamente em partes relevantes da entrada ao processar informações. Ao invés de tratar todas as palavras igualmente, ele aprende quais *tokens* são mais relacionados entre si na base de treinamento e atribui um peso correspondente.

A Figura 2.3 representa a implementação de atenção de [Vaswani et al., 2017]. A atenção, como descrita pelo autor, é calculada com a fórmula a seguir, onde **Q** é a matriz de *queries*, derivada dos *embeddings* de entrada, **K** é a matriz de *keys*, também derivada dos *embeddings*, **V** é a matriz de *values*, que contém as informações associadas aos *tokens* e **d** é o número de dimensões dos vetores de chave:

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

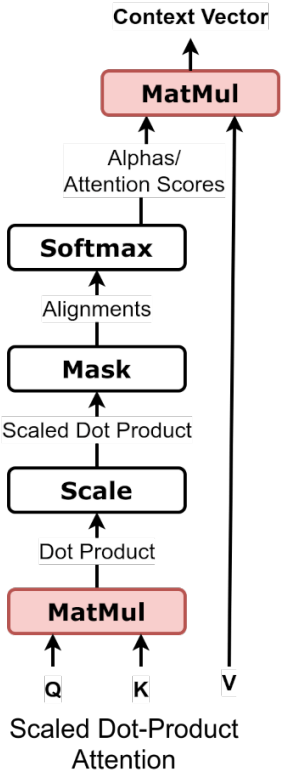

Figura 2.3: Diagrama de Scaled Dot-Product Attention. Imagem de [dvgodoy, 2025]

Já o *self-attention* é um tipo especializado de atenção utilizado na arquitetura *transformer* que, ao invés de comparar duas sequências diferentes para descobrir a relação entre elas como é feito na atenção tradicional, compara a sequência com ela mesma. Essa estratégia permite o treinamento com dados pouco ou não estruturados, como os textos encontrados na internet.

Para potencializar a captura de diferentes padrões de relacionamento entre palavras, o *transformer* utiliza várias "cabeças" de atenção simultaneamente (*Multi-Head self-Attention*), como pode ser visualizado na Figura 2.4. O cálculo é feito dividindo os embeddings em h cabeças e aplicando diferentes projeções para cada uma, como está representado na fórmula abaixo:

$$\text{MultiHead}(Q, K, V) = \text{Concatenar}(\text{cabeça}_1, \ldots, \text{cabeça}_h)\, W^O$$

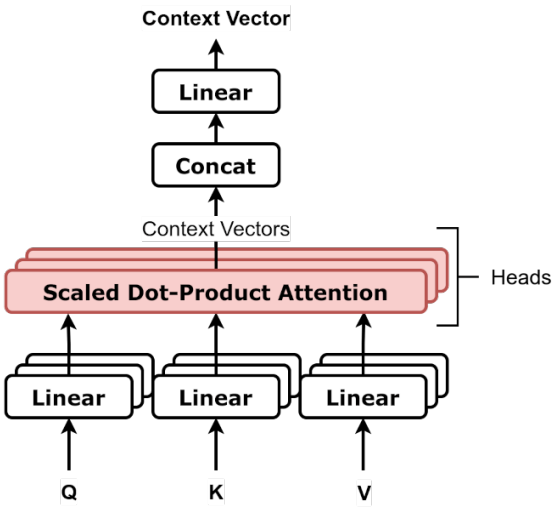

Figura 2.4: Diagrama de Multi-Head Attention. Imagem de [dvgodoy, 2025]

Feed-Forward Networks

Os embeddings, após passarem pela camada de *multi-headed self-atention*, são novamente normalizados. O vetor de *embedding* de cada *token* então é submetido separadamente a uma *feed-forward network*, que são um tipo de rede neural onde a informação flui apenas em uma única direção, da camada de entrada para a camada de saída, sem ciclos ou conexões recorrentes [Sazli, 2006]. A rede tem a função de refinar a representação dos embeddings, permitindo ao modelo capturar relações complexas entre as palavras. O *feed-forward network* é aplicado de forma independente a cada posição do embedding e é idêntico em todas as posições, a *feed-forward network* em Transformers é composta por duas camadas lineares separadas por uma função de ativação não linear. Ele pode ser definido por

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2$$

Onde x é a entrada da camada da *feed-forward network*, que vem da saída do mecanismo de atenção, W_1 e W_2 são matrizes de pesos das duas camadas lineares, b_1 e b_2 são os vieses correspondentes, e a função $max(0, x)$ representa a ativação ReLU. O resultado, então, é novamente normalizado e enviado para as próximas camadas do *transformer*.

2.3 Retrieval-Augmented Generation (RAG)

O *Retrieval-Augmented Generation* (RAG) [Lewis et al., 2020] é uma abordagem que combina técnicas de recuperação de informações com modelos generativos, como os LLMs.

Enquanto os modelos de linguagem são poderosos para gerar texto coerente, eles podem não ser confiáveis em termos de precisão factual. O RAG aborda esse problema ao integrar um mecanismo de recuperação de documentos que fornece informações relevantes para o modelo generativo durante a fase de inferência. A estrutura geral do RAG pode ser vista na Figura 2.5.

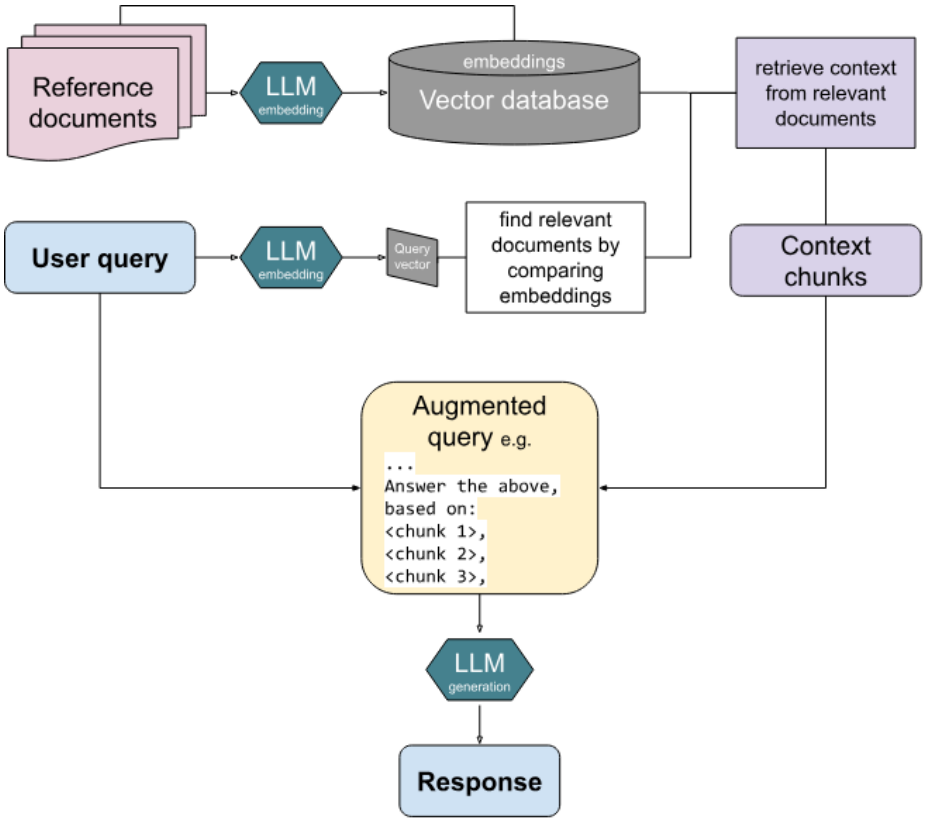

Figura 2.5: Diagrama de Fluxo Geral RAG. Imagem de [Commons, 2024]

A abordagem RAG funciona em dois estágios principais :

- **Recuperação de Dados**: Um *retriever* busca documentos relevantes em uma base de dados, podendo se utilizar de técnicas de similaridade semântica baseadas em *embeddings*.
- **Geração de Respostas**: O modelo generativo usa os documentos recuperados como contexto adicional para produzir respostas informadas e precisas.

Essa combinação é particularmente útil em domínios sensíveis, como o da saúde, onde a confiabilidade das informações é crucial. O RAG também permite que o sistema se adapte rapidamente a novos dados, tornando-o uma solução flexível para aplicações práticas. [Lewis et al., 2020]

2.3.1 Recuperação de dados (retrieval)

A recuperação de informações é um componente essencial no RAG, responsável por buscar documentos relevantes em uma base de conhecimento antes que o modelo de linguagem gere uma resposta.

Recuperação Baseada em Palavras-chave (Sparse Retrieval)

Os métodos tradicionais de recuperação de informações são baseados na correspondência exata de termos, sem levar em conta relações semânticas mais profundas entre palavras, sendo possível exemplificar o BM25 [Robertson and Zaragoza, 2009].

O BM25 é uma evolução do modelo TF-IDF , avaliando a relevância de um documento d para uma consulta q atribuindo um escore baseado na frequência dos termos e no comprimento do documento, baseado na seguinte fórmula:

$$\text{score}(D, Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{\text{avgdl}})}$$

Onde $f(q_i, D)$ é frequência do termo q_i no documento D, a mesma do TF-IDF, $|D|$ é o comprimento do documento, **avgdl** é o comprimento médio dos documentos na coleção, **k e b** são parâmetro de suavização e normalização e **IDF** é o Fator de Inversão de Frequência do Documento, nesse caso é uma variente da do TF-IDF, havendo a adição de 0.5 no numerador e denominador do logarítmo.

Dense Retrieval e indexação vetorial

Os métodos de *dense retrieval* utilizam representações vetoriais de alta dimensionalidade para capturar melhor o significado semântico das palavras e das sentenças. No dense retrieval há o uso de modelos de embedding para transformar consultas e documentos em representações vetoriais. Após converter documentos e consultas em representações vetoriais, é necessário um método eficiente para recuperar os documentos mais relevantes.

Para isso, utilizam-se estruturas e ferramentas de indexação vetorial como :

- **FAISS**[2] **(Facebook AI Similarity Search)**: Biblioteca otimizada para busca rápida de vetores de alta dimensão [Johnson et al., 2021].
- **ScaNN**[3] **(Scalable Nearest Neighbors)**: Biblioteca desenvolvida pelo Google focada em eficiência para grandes bases de embeddings [Guo et al., 2020]

[2] `https://github.com/facebookresearch/faiss`
[3] `https://github.com/google-research/google-research/tree/master/scann`

A eficiência da recuperação vetorial depende da métrica utilizada para medir similaridade, sendo possível exemplificar a Similaridade Cosseno, que mede o ângulo entre dois vetores no espaço vetorial, e a Distância Euclidiana, que mede a proximidade entre os vetores no espaço, considerando a diferença ponto a ponto entre seus componentes. Em ambos os casos, A e B são os embeddings a serem comparados, por exemplo, o *embedding* de uma *query* que um usuário realizou com o *embedding* de um *chunk* de documento.

A similaridade cosseno é definida pela fórmula a seguir, onde $\vec{A}$ e $\vec{B}$ são os vetores serem comparados, $\vec{A} \cdot \vec{B}$ é o produto escalar entre os vetores e $\|\vec{A}\|\|\vec{B}\|$ são as normas dos respectivos vetores.

$$\text{Similaridade Cosseno (cosine)} = \frac{\vec{A} \cdot \vec{B}}{\|\vec{A}\|\|\vec{B}\|}$$

A distância euclidiana é definida pela fórmula a seguir, onde A_i e B_i são as componente dos vetores na dimensão i.

$$\text{Distância Euclidiana (l2)} = \sqrt{\sum_{i=1}^{n}(A_i - B_i)^2}$$

Grafo de conhecimento

O GraphRAG expande o paradigma tradicional do RAG ao incorporar grafos de conhecimento como fonte primária de recuperação de informações. Ao contrário dos métodos convencionais, que dependem de documentos textuais não estruturados, o GraphRAG utiliza a estrutura dos grafos para capturar relações complexas entre entidades, permitindo uma recuperação mais precisa e uma geração de respostas mais coerente [Han et al., 2025].

A arquitetura da GraphRAG pode ser dividida em três etapas principais: Indexação Baseada em Grafos, Recuperação Guiada por Grafos e Geração Aprimorada por Grafos [Peng et al., 2024].

1. **Indexação Baseada em Grafos**: Nesta etapa, os dados são estruturados em um grafo, onde as entidades são representadas como nós e suas inter-relações como arestas.

2. **Recuperação Guiada por Grafos**: Com a estrutura de grafo estabelecida, a recuperação de informações é orientada pelas conexões entre os nós.

3. **Geração Aprimorada por Grafos**: Na fase de geração, o modelo utiliza as informações recuperadas, juntamente com a estrutura do grafo, para produzir respostas que consideram o contexto relacional das entidades envolvidas.

2.3.2 Geração (Generation)

A etapa de geração em um sistema RAG é responsável por transformar as informações recuperadas em uma resposta coerente e relevante para o usuário. Diferente de sistemas tradicionais de recuperação de informação, nos quais o usuário recebe diretamente documentos ou trechos de texto, um modelo RAG combina a busca com um LLM para gerar uma resposta textual baseada nos documentos relevantes.

Modelos de Geração

A etapa de geração em RAG costuma ser conduzida por modelo baseado na arquitetura Transformer, dentre os quais é possível citar modelos como o GPT-4 [OpenAI et al., 2024], T5 [Raffel et al., 2020] e Llama3 [Grattafiori et al., 2024], o qual foi utilizado para o desenvolvimento da aplicação.

Ollama e Llama3

Large Language Model Meta AI 3 (Llama3) é a terceira iteração da série de modelos de linguagem desenvolvidos pela Meta AI [Touvron et al., 2023]. Esses modelos foram projetados para fornecer desempenho avançado em uma ampla gama de tarefas de PLN, incluindo tradução, classificação de texto, resumo e geração de linguagem natural.

O uso do Llama3, é feito com intermédio do Ollama, que é uma plataforma que visa facilitar a implementação e utilização de modelos de linguagem avançados em ambiente local, sem dependência de serviços baseados na nuvem. Esse enfoque é especialmente relevante para organizações que possuem preocupações com privacidade, confidencialidade de dados e autonomia tecnológica.

A arquitetura da plataforma Ollama é projetada para integrar modelos de linguagem com eficiência em dispositivos locais, utilizando técnicas avançadas de otimização de hardware, como paralelismo de GPU e redução de precisão em cálculos (*quantization*) . Isso permite que modelos robustos, como GPT ou Llama, sejam utilizados em sistemas com recursos computacionais limitados.

2.3.3 Avaliação

Para ajustar parâmetros, realizar o retreino e verificar se o modelo está pronto para uso no mundo real, é fundamental adotar um método de avaliação de desempenho. Ao longo do tempo, diversos métodos, tanto automatizados quanto manuais, foram desenvolvidos para essa finalidade. Nesta subseção, será discutido alguns desses métodos.

Avaliação Humana

A avaliação humana é um método robusto para avaliar a qualidade de modelos de linguagem generativos, dado que a metrificação com o usuário final permite uma avaliação direta da qualidade, apesar de apresentar um alto de custo de execução e possíveis vieses amostrais. Embora métodos automatizados possam servir como apoio para direcionar e acelerar o processo, a análise humana apresenta-se como uma boa ferramenta para assegurar a qualidade dos resultados.

Bilingual Evaluation Understudy (BLEU)

A métrica BLEU [Papineni et al., 2002] foi criada inicialmente para avaliar traduções, comparando a tradução de referência com a tradução gerada por um modelo de linguagem. O método pode ser utilizado na avaliação de RAGs ao comparar a resposta gerada pelo modelo com a resposta original.

O BLEU mede a similaridade entre dois textos utilizando n-gramas. Ele compara quantos n-gramas vindos da saída do modelo também aparecem na referência. Além disso ele também penaliza diferenças de tamanho muito grandes.

O método possui uma precisão próxima ao julgamento humano[Papineni et al., 2002] para traduções e sinônimos considerando seu baixo custo de computação, mas tem algumas limitações. Por fazer uma comparação exata de palavras, o método não consegue detectar sinônimos nem parafraseamento. Além disso ele mede apenas semelhança entre as palavras dos dois textos, não verificando a factualidade do conteúdo gerado.

Recall-Oriented Understudy for Gisting Evaluation(ROUGE)

A métrica ROUGE [Lin, 2004] foi criada inicialmente para a avaliação de qualidade de resumos de texto. No contexto do RAG, ela mede a similaridade entre a resposta de resposta de referência e a resposta gerada pelo modelo.

O ROUGE mede quantos n-gramas vindos da resposta de referência encontram-se também na resposta gerada pelo modelo. O método possui uma precisão bem satisfatória considerando seu baixo custo de computação, mas tem algumas limitações. Por fazer uma comparação exata de palavras e sequências de palavras, não consegue detectar sinônimos nem parafraseamento.

Enquanto o BLEU mede se o conteúdo da resposta gerada é coerente com a resposta de referência, o ROUGE busca identificar se alguma parte da resposta de referência não foi contemplada na resposta gerada pelo modelo. Dessa forma as duas métricas se complementam.

BertScore

Diferente do Bleu e do Rouge, que fazem comparação exata de n-gramas, o BertScore [Zhang et al., 2020] compara a representação vetorizada. Utiliza o modelo BERT para gerar os embeddings da resposta gerada e da resposta de referência e calcula a similaridade.

A similaridade é calculada usando algum método como distância de cosseno. Por conta dessa comparação, o método consegue detectar sinônimos e variações no texto.

Ao final da comparação, o BertScore retorna 3 valores:

- Precision: Mede a proporção de tokens do texto gerado que possuem correspondências na referência, indicando o quanto do texto gerado é relevante.
- Recall: Avalia a proporção de tokens da referência que são encontrados no texto gerado, refletindo o quanto da referência foi capturado.
- F1-score: Média harmônica entre precision e recall, equilibrando ambas as métricas para fornecer uma avaliação geral da similaridade.

2.3.4 Frameworks de avaliação automatizada

Existem alguns frameworks que juntam diversas métricas para avaliação de modelos de linguagem e facilitam o processo de automação. Alguns desses frameworks são discutidos nessa subseção, com uma maior ênfase no RAGAS[4], que foi o framework utilizado neste trabalho.

ReEval

ReEval [Yu et al., 2024b] É um framework de avaliação inspirado no conceito de aprendizado adversarial. Ele utiliza um LLM para fazer ataques adversariais para criar cenários de teste dinâmicos, gerando novos contextos a partir de exemplos estáticos. Essa abordagem permite avaliar a capacidade dos modelos de evitar alucinações de uma forma que apenas casos de teste estáticos não seriam capazes de fazer.

Além disso, os exemplos adversariais gerados podem ser reutilizados testes com diferentes LLMs. LLMs menos poderosas podem ser utilizadas para gerar testes para LLMs mais poderosas, o que torna a abordagem bem eficiente em termos de custos.

TruLens

TruLens [TruEra, 2025] é uma ferramenta de código aberto desenvolvida pela empresa TruEra. É utilizada para medir desempenho de aplicações baseadas em LLMs, como

[4] `https://github.com/explodinggradients/ragas`

RAGs. Mede características como relevância, precisão, confiabilidade e fidelidade, assim como ajuda a identificar alucinações.

O maior diferencial da ferramenta é a capacidade de monitorar em tempo real o desempenho dos modelos, permitindo a criação de benchmarks personalizados pelo usuário. Essa flexibilidade torna a ferramenta ideal para domínios de conhecimento muito especializados, como a medicina e o direito.

RAGAS

O framework RAGAS [Es et al., 2024] foi desenvolvido para avaliar a qualidade de sistemas de Geração Aumentada por Recuperação (RAG), oferecendo um conjunto abrangente de métricas que analisam a relação entre a consulta do usuário, o contexto recuperado e a resposta gerada pelo modelo. Sua abordagem visa superar as limitações das avaliações tradicionais de RAG, permitindo uma análise mais estruturada e automatizada do desempenho desses sistemas por meio de um foco em avaliação baseada em LLM.

Para o estudo, o RAGAS foi escolhido por ser um framework com bastante utilização e integrações com outras bibliotecas, tendo como base outros estudos como [Roychowdhury et al., 2024] no qual há a utilização do RAGAS para avaliar e comparar diferentes modelos em um sistema RAG.

O framework RAGAS possui várias métricas para determinar a avaliação de sistemas, abaixo estão as descrições das métricas utilizadas nesse trabalho, como especificado na documentação oficial do Ragas [Ragas, 2025]. O processo de avaliação de todas as métricas funciona a partir de duas ou mais entre essas quatro entradas:

1. **Pergunta original**: Pergunta realizada ao sistema RAG.
2. **Resposta original**: Resposta correta de referência utilizada como modelo de verdade.
3. **Resposta gerada pelo RAG**: Resposta que o RAG gerou com base na pergunta original e no contexto recuperado.
4. **Contexto recuperado pelo RAG**: Seções mais relevantes de documentos da base vetorial encontradas pelo RAG com base na pergunta original.

Semantic Similarity

A métrica *Semantic Similarity* calcula a semelhança semântica entre a resposta gerada pelo RAG e a resposta original da pergunta. Seu valor vai de -1 a 1, mas na maior parte das vezes será um valor entre 0 e 1. Um valor maior significa maior alinhamento semântico entre as respostas comparadas.

A métrica é calculada em três passos:

1. Vetorizar a resposta original utilizando um embedding.
2. Vetorizar a resposta gerada pelo RAG utilizando o mesmo embedding.
3. Calcular a similaridade de cosseno entre os dois vetores

Faithfulness

A métrica *Faithfulness* mede o quão consistente a resposta gerada pelo RAG é com o contexto recuperado. Uma resposta "faithfull", ou seja, fiel ao contexto, é aquela em que todas as afirmações contidas na resposta podem ser derivadas do contexto.

Ela é calculada da seguinte maneira:

1. Identificar todas as afirmações contidas em uma resposta. Essa etapa é realizada por um LLM.
2. Checar se cada afirmação pode ser inferida do contexto. Pode ser feito utilizando um LLM ou outro modelo de comparação. Nesse trabalho foi utilizado um LLM.
3. Calcular a métrica utilizando a fórmula abaixo.

$$\text{Faithfulness} = \frac{\text{Numero de afirmações na resposta suportadas pelo contexto}}{\text{Número total de afirmações na resposta}}$$

Answer Relevancy

A métrica *Answer Relevancy* mede o quão relevante é a resposta em relação à pergunta feita pelo usuário. Uma pontuação baixa pode indicar que a resposta está incompleta ou contém informação redundante. A métrica é calculada em três passos:

1. É utilizada uma LLM para gerar uma lista de perguntas baseado na resposta, tentando supor qual era a pergunta original. O padrão é criar 3 perguntas.
2. É computada a similaridade entre cada pergunta gerada (E_{g_i}) e a pergunta original (E_o).
3. É feita a média entre as similaridades calculadas.

$$\text{Answer Relevancy} = \frac{1}{N}\sum_{i=1}^{N} \text{similaridade de cosseno}(E_{g_i}, E_o)$$

Recall de Contexto em Sistemas Baseados em LLM

A métrica *Recall de Contexto* avalia a eficácia na recuperação de documentos ou informações relevantes. Os valores variam de 0 a 1, sendo que valores mais altos indicam melhor desempenho. É calculada utilizando os seguintes passos:

1. Quebrar a resposta original em afirmações utilizando uma LLM
2. Cada afirmação é analisada para determinar se ela pode ser inferida de algum contexto recuperado
3. O resultado final é calculado utilizando a fórmula abaixo:

$$\text{Context Recall} = \frac{\text{Número de afirmações da resposta que podem ser inferidos do contexto}}{\text{Número total de afirmações da resposta}}$$

Context Precision

A métrica *Context Precision* mede a proporção de chunks relevantes no contexto recuperado. É feita a comparação entre cada chunk e a resposta original. Essa avaliação de relevância pode ser feita utilizando um LLM ou outros métodos, como similaridade de cosseno. Neste trabalho, foi utilizado um LLM.

Answer Correctness

A métrica *Answer Correctness* compara a resposta gerada pelo modelo com a resposta original. Ela é uma média ponderada entre *semantic similarity* e *factual correctness*. *Semantic similarity* é a distância de cosseno entre a resposta gerada e a original, já *factual similarity* é a sobreposição de fatos entre a resposta gerada pelo RAG e a original. O Ragas utiliza uma LLM para definir quais afirmações presentes na resposta geradas estão corretas de acordo com a resposta original. As afirmações são classificadas em verdadeiros positivos (VP), falsos positivos (FP) e falsos negativos (FN) pelo modelo utilizado com base nos seguintes crtitérios:

- **VP**: Fatos ou afirmações que estão presentes tanto resposta original quanto na resposta gerada.
- **FP**: Fatos ou afirmações que estão presentes na resposta gerada, mas não na resposta original.
- *FN*: Fatos ou afirmações que estão presentes na resposta original, mas não na resposta gerada.

Por fim, esses valores são utilizados para calcular um *F1 score*, seguindo a fórmula abaixo.

$$\text{F1-score de Factual Correctness} = \frac{|\text{VP}|}{|\text{VP}| + 0.5 \times (|\text{FP}| + |\text{FN}|)}$$

O valor final da métrica Answer Correctness é calculado por uma média ponderada em que factual correctness representa 75% e semantic similarity representa 25%, podendo haver alteração nos pesos caso o usuário deseje.

2.4 Conclusão do Referencial Teórico

O referencial teórico abordou os principais conceitos relacionados a LLMs, PLN e transformers, fornecendo uma base sólida para a compreensão do funcionamento dos sistemas baseados em RAG. Foram discutidas também diferentes abordagens para a recuperação de informações, incluindo métodos tradicionais, como BM25, e técnicas mais recentes, como GraphRAG, que integram grafos de conhecimento para melhorar a precisão e a contextualização das respostas geradas.O estudo dessas abordagens reforça a necessidade de experimentação e otimização contínuas para aprimorar a aplicabilidade desses sistemas, especialmente em domínios sensíveis, como a saúde. Com essa base teórica, este trabalho segue para a etapa metodológica, onde serão detalhados os processos de implementação e avaliação do chatbot RAG.

Capítulo 3

Metodologia parte 1: Desenvolvimento da aplicação

A metodologia foi dividida em duas partes. A primeira parte, desenvolvida neste capítulo, trata do processo de desenvolvimento do *chatbot* utilizando RAG. A segunda parte, desenvolvida no próximo capítulo, trata da avaliação do desempenho do *chatbot*.

Este capítulo é dividido em 3 seções. As duas primeiras tratam do RAG, sendo a primeira sobre a etapa de *retrieval* e a segunda sobre a etapa de *generation*. A última seção trata da interface com o usuário por meio de aplicativo de mensagem.

A figura 3.1 representa a estrutura geral da aplicação desenvolvida.

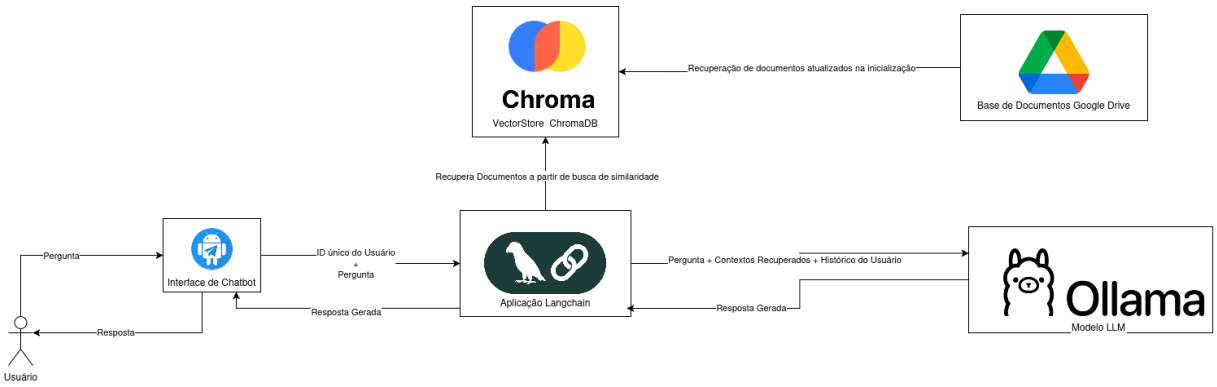

Figura 3.1: Diagrama da aplicação

A principal tecnologia utilizada no desenvolvimento da aplicação foi a linguagem de programação *Python* e todo o código fonte está disponibilizado no github[1].

Para toda a computação intensiva e dependente de GPU realizada durante o trabalho, foi utilizada uma máquina compartilhada fornecida pelo Departamento de Ciência

[1] `https://github.com/Anon1929/tcc_rag_saude`

da Computação da Universidade de Brasília. A máquina conta com as seguintes configurações:

- Processador: Intel(R) Core(TM) i5-9400F CPU @ 2.90GHz (6 núcleos)
- Memória RAM: 32 GB
- GPU: GeForce RTX 3060 12GB de VRAM

3.1 Retrieval

Retrieval é a parte do RAG responsável por trazer informações relevantes para auxiliar a geração de respostas. Nesta seção é definida a fonte dos dados utilizados, o tratamento desses dados e de como essa informação é buscada e disponibilizada para as próximas etapas do processo.

3.1.1 Fonte dos dados

A base de conhecimento utilizada pelo *chatbot* foi construída em cima de 13 **cadernos de atenção básica** disponibilizados no site de **ministério da saúde** [2]. Os Cadernos de Atenção Básica são documentos técnicos que orientam profissionais do Sistema Único de Saúde (SUS) sobre a organização e a qualificação da atenção primária à saúde no Brasil. Essas diretrizes foram escolhidas com base em recomendações e seleções da docente Anna Carolina Faleiros Martins[3] dada a área correlata de atuação.

Os documentos tratam de diversos temas, como: saúde da criança, do idoso e da mulher, doenças crônicas, saúde mental, estratégia saúde da família, promoção da saúde e prevenção de doenças. Os cadernos estão disponibilizados no formato PDF e cada caderno possui cerca de 180 páginas.

Os PDFs em questão estavam bem estruturados, por isso não foi necessária a utilização de técnicas de extração mais complexas, como reconhecimento óptico de caracteres. Foi utilizada a biblioteca PyPDFLoader para realizar a extração do texto bruto desses documentos.

3.1.2 Partição do texto

Após ser extraído, o conteúdo bruto dos documentos foi separado em fragmentos. O objetivo dessa separação é criar fragmentos semanticamente coerentes. Buscando atingir esse

[2] `https://bvsms.saude.gov.br/diretrizes/`
[3] `https://sigaa.unb.br/sigaa/public/docente/portal.jsf?siape=2007577`

objetivo, utilizou-se *textSplitters*, que são algoritmos que dividem um texto em fragmentos, seguindo uma regra predefinida, que deve ser configurada pelo usuário de acordo com o uso desejado. Foi utilizado dois *textSplitters* fornecidos pela biblioteca Langchain[4]:

CharacterTextSplitter

Um divisor extremamente simples, projetado para dividir um texto sempre que encontrar um separador definido pelo usuário. Esse separador pode ser um único caractere ou uma sequência de caracteres. A Figura 3.2 mostra visualmente o funcionamento do *splitter*, em que há uma quebra de texto simples.

Por padrão, o *CharacterTextSplitter* utiliza a sequência "$\backslash n\backslash n$", que corresponde a uma linha em branco no texto. Contudo, o separador não foi incluído no experimento para esse *textSplitters* devido ao fato de sua utilização resultar em *chunks* de tamanhos excessivos, ultrapassando o limite do modelo de *embeddings* e trazendo resultados negativos sem relevância, dessa forma, para este trabalho os seguintes separadores foram experimentados:

- "$\backslash n$"
- "."

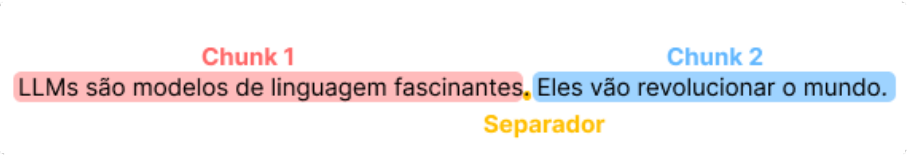

Figura 3.2: CharacterTextSplitter

RecursiveCharacterTextSplitter

Ao invés de dividir o texto com base em apenas um separador, esse método permite o uso de uma lista de separadores. O usuário define um tamanho desejado para cada trecho, chamado *chunk size*. O *splitter*, então, segmenta o texto utilizando o primeiro separador, seguido pelo segundo e assim por diante, até que cada trecho gerado tenha um tamanho menor ou igual ao especificado. A Figura 3.3 mostra visualmente o funcionamento do *splitter*.

As seguintes listas de separadores foram experimentadas:

- "$\backslash n\backslash n$" e "$\backslash n$"
- "$\backslash n\backslash n$", "$\backslash n$" e "."

[4] `https://www.langchain.com/`

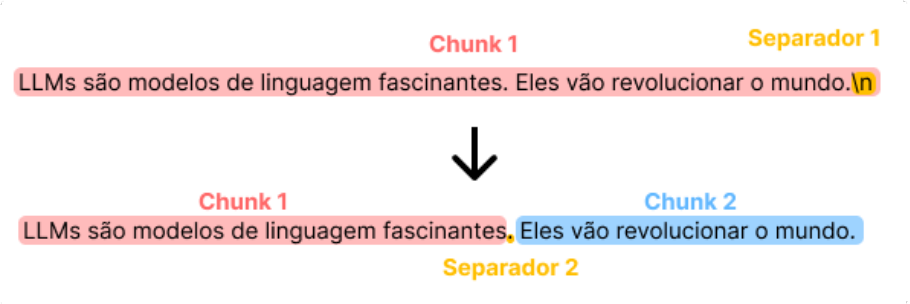

Figura 3.3: RecursiveCharacterTextSplitter

Esse *splitters* também permite a sobreposição entre dois trechos, ou seja, a porção final de um trecho é repetida no próximo, como pode ser visualizado na Figura 3.4. Essa técnica pode ajudar a manter a continuidade do contexto e melhorar a recuperação de informações. O tamanho dessa sobreposição é chamado de *chunk overlap*.

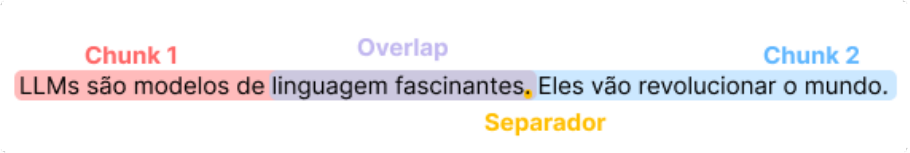

Figura 3.4: Chunk Overlap

Esses *splitters* foram escolhidos pelo fato de funcionam bem para texto pouco estruturado, como é o caso da base escolhida. O custo computacional dos mesmos também é bem pequeno, por se tratarem de regras simples de separação.

3.1.3 VectorStore

Os fragmentos foram, em seguida, transformados em vetores de *embeddings*. O modelo de *embedding* escolhido foi o *intfloat/multilingual-e5-large*[5], compatível com a implementação *FastEmbedEmbeddings* [6] no LangChain. Esse modelo foi selecionado por sua característica de ser multilíngue e pela capacidade de processar entradas maiores, de até 1024 *tokens*, atributos que se mostraram cruciais para documentos técnicos extensos e diversificados.

Esses vetores, então, foram armazenados em um banco vetorial criado utilizando o *ChromaDB*[7]. A ferramenta fornece uma infraestrutura otimizada para recuperação de informações baseadas em similaridade de vetores, permitindo buscas eficientes.

[5] `https://huggingface.co/intfloat/multilingual-e5-large`

[6] `https://python.langchain.com/api_reference/community/embeddings/langchain_community.embeddings.fastembed.FastEmbedEmbeddings.html`

[7] `https://www.trychroma.com/`

Ao realizar uma busca, o usuário do pode definir um valor mínimo de similaridade para o *chunk* poder ser incluído no resultado, bem como o número máximo de *chunks* a serem retornados. A regulação desses parâmetros representa o balanceamento entre volume de contexto recuperado e ruído.

3.2 Generation

A etapa de *Generation* no RAG é responsável por gerar a resposta final com base no contexto recuperado pelo *Retrieval*. Nesta seção, é abordada a integração entre a LLM e o *retriever*, além das técnicas empregadas para aprimorar a qualidade das respostas geradas.

A integração do RAG foi realizada com o uso da *RetrievalQA Chain* do LangChain, que é especializada em responder perguntas sobre um contexto. Ela recebe como argumento o LLM a ser utilizado, o *prompt* e o *retriever*.

LangChain é uma biblioteca projetada para construir aplicações baseadas em LLMs. Ela facilita a integração de LLMs com outras fontes de dados, como bancos de dados, *APIs* e fluxos de trabalho personalizados, sendo amplamente utilizada no desenvolvimento de sistemas RAG. A arquitetura modular do LangChain permite a construção de sistemas complexos, dividindo a funcionalidade em componentes reutilizáveis.

3.2.1 Integração com LLM

O modelo de linguagem integrado ao sistema foi o Llama3, que conta com 8 bilhões de parâmetros e tem uma janela de contexto de 8192 *tokens*. A escolha desse modelo levou em consideração tanto a qualidade da geração quanto a eficiência no uso de recursos, alinhando-se às limitações de hardware da máquina disponível.

A integração com o LLM foi realizada por meio da plataforma Ollama, que fornece uma interface robusta e fácil de usar para gerenciamento modelos de linguagem. O Ollama permite a execução local de modelos, oferecendo suporte para otimização de uso de recursos e melhoria de desempenho.

3.2.2 Prompt

Foi utilizado um *prompt* cuidadosamente projetado, com foco na clareza, e estruturado de maneira a guiar o modelo na geração de respostas relevantes. Além disso, foram incluídas instruções específicas com o objetivo de minimizar alucinações e respostas não baseadas no contexto recuperado.

O *prompt* utilizado foi o seguinte:

```
Você é um assistente virtual treinado para ajudar agentes comunitários de saúde
do SUS (Sistema Único de Saúde) a realizarem suas tarefas diárias.
Seu principal objetivo é fornecer orientações precisas baseadas nas diretrizes
de saúde estabelecidas, responder perguntas e manter conversas informativas
relacionadas à saúde.
É crucial que você não forneça informações não verificadas ou alucinações além
do escopo de saúde.
Instruções:
Foco na Saúde:
    Responda somente a perguntas e dê orientações relacionadas à saúde.
    Use informações das diretrizes de saúde estabelecidas como base para suas respostas.
Concisão e Clareza:
    Forneça respostas claras e concisas.
    Evite jargões técnicos que possam confundir o agente comunitário.
Atenção aos Detalhes:
    Preste atenção aos sintomas, condições e perguntas específicas apresentadas
    pelo agente.
    Ofereça recomendações práticas e orientações de acompanhamento conforme necessário.
Evite Alucinações:
    Não invente informações ou dados. Se não souber a resposta, oriente o agente
    a buscar fontes confiáveis ou consultar um profissional de saúde.
Conversa Natural:
    Mantenha um tom amigável e profissional, semelhante a uma conversa com um colega.
    Esteja sempre disposto a ajudar e apoiar o agente comunitário em suas tarefas.
Use os pedaços de contexto a seguir para responder a pergunta no final.
Responda sempre em português.
{context}
Pergunta: {question}
Resposta útil:
```

3.2.3 Enriquecimento da resposta gerada

Como medida adicional para assegurar a veracidade das respostas, a saída gerada pelo modelo foi enriquecida com a indicação do documento que originou o contexto da resposta a partir dos metadados, comportamento observável na Figura 3.7. Essa funcionalidade aumenta a confiabilidade do *chatbot*, permitindo que os usuários validem as informações fornecidas e identifiquem rapidamente a origem dos dados.

Adicionalmente, foi implementada uma verificação programática que avalia a quantidade de fragmentos (*chunks*) recuperados no contexto. Caso o número de fragmentos

recuperados seja igual a zero, a resposta do RAG é substituída por uma mensagem informando a inexistência de documentos relevantes.

3.3 Comunicação com usuário final

Para facilitar a utilização e o entendimento do produto pelos usuários finais, os agentes de saúde, foi escolhido o modelo de chat em aplicativo de mensagem. O aplicativo de mensagem escolhido foi o Telegram devido à sua gratuidade e farta disponibilização de ferramentas que facilitam a criação de *chatbots*.

Para registro e configuração do *chatbot*, foram utilizados os recursos oferecidos pela ferramenta *BotFather*[8], disponibilizada oficialmente pelo Telegram. Com o *BotFather* é possível criar, personalizar e gerenciar *bots* do telegram para implementações com api, sendo possível observar um exemplo de fluxo de criação de *tokens* de *API* pela Figura 3.5.

[8] `https://telegram.me/BotFather`

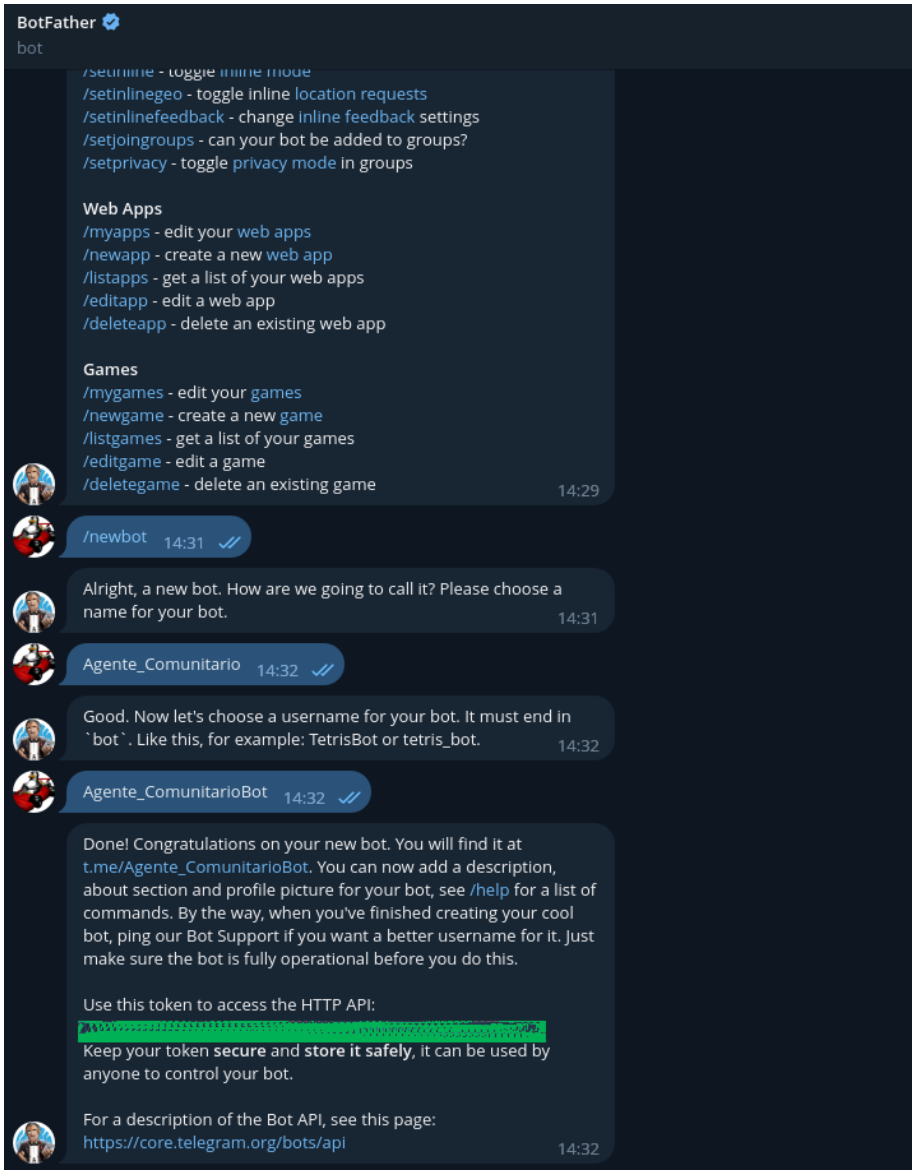

Figura 3.5: Exemplo de criação de *tokens* para *bot*.

A comunicação do bot com a aplicação foi realizada com a biblioteca *python-telegram-bot*[9], a qual implementa a API original do Telegram facilitando o interfaceamento com o python, feito por meio do método de *polling*, que permite que o sistema busque por atualizações de mensagens recebidas de maneira assíncrona por requisições ao servidor central.

Para o usuário iniciar uma conversa com o *chatbot*, é possível acessar o link[10] ou iniciar uma busca na barra de pesquisa do Telegram pelo *username* cadastrado, no caso do *bot* do estudo **Agente_comunitarioBot**.

[9] `https://docs.python-telegram-bot.org/en/v21.10/`
[10] `https://t.me/Agente_ComunitarioBot`

O componente de memória do LangChain foi utilizado para manter o histórico de interações dos usuários. Esse histórico é organizado em um dicionário python, onde cada entrada é identificada pelo ID do usuário no Telegram. Isso permite que o *chatbot* tenha o contexto de cada conversa, mesmo com diversos usuários utilizando a ferramenta simultaneamente.

As figuras 3.6 e 3.7 mostram exemplos reais de conversa com o *chatbot*.

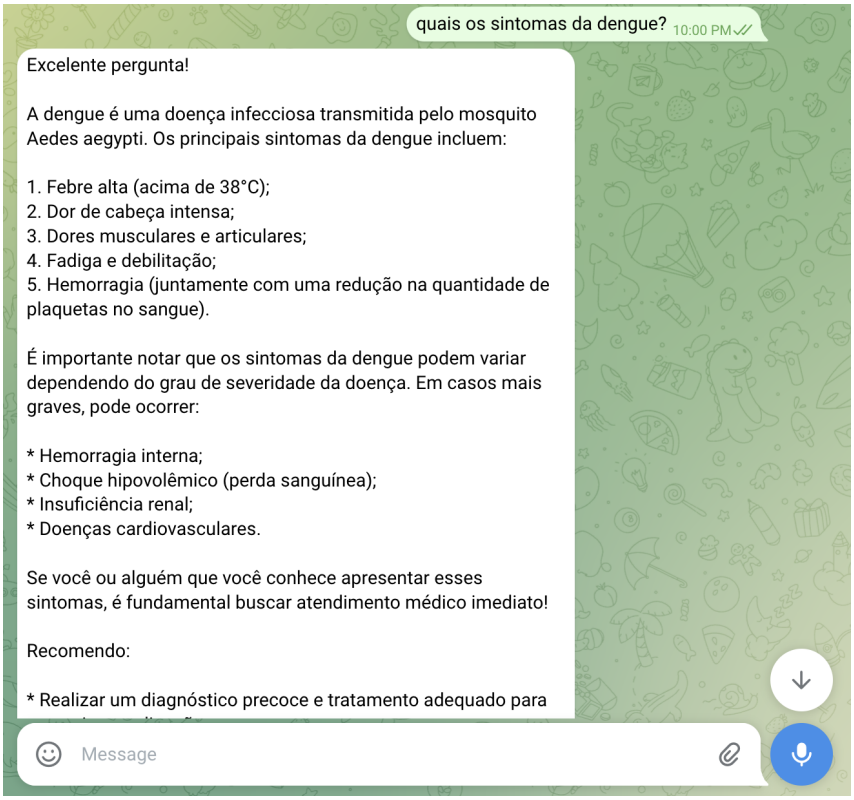

Figura 3.6: Print Telegram

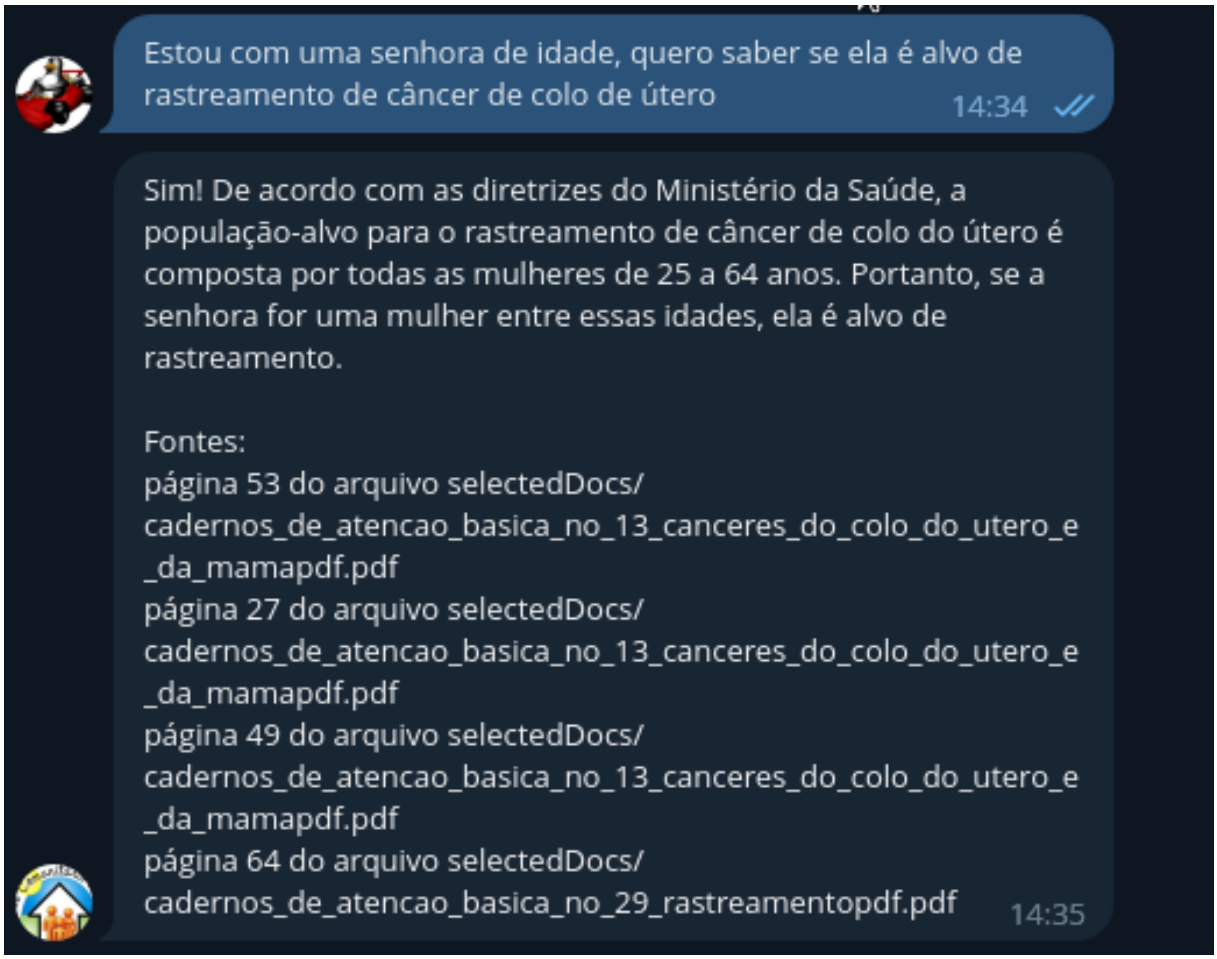

Figura 3.7: Exemplo de conversa com *chatbot*

Capítulo 4

Metodologia parte 2: Avaliação

A avaliação do sistema RAG foi conduzida com o objetivo de mensurar a qualidade das respostas geradas pelo *chatbot* e compreender o impacto das diferentes configurações de segmentação de texto, funções de distância e *overlaps* de *chunks*.

Inicialmente, foi planejada uma abordagem de avaliação manual, porém, devido a limitações logísticas, optou-se por uma avaliação automatizada utilizando o *framework RAGAS*.

Para isso, foi desenvolvido um *fluxo* de avaliação composto pelas quatro etapas principais enumeradas abaixo, que são descritas nas seções deste capítulo:

1. Geração de base de perguntas e respostas.
2. Construção dos *vectorStores*
3. Utilização do RAG para responder às perguntas da base.
4. Avaliação dos resultados utilizando RAGAS.

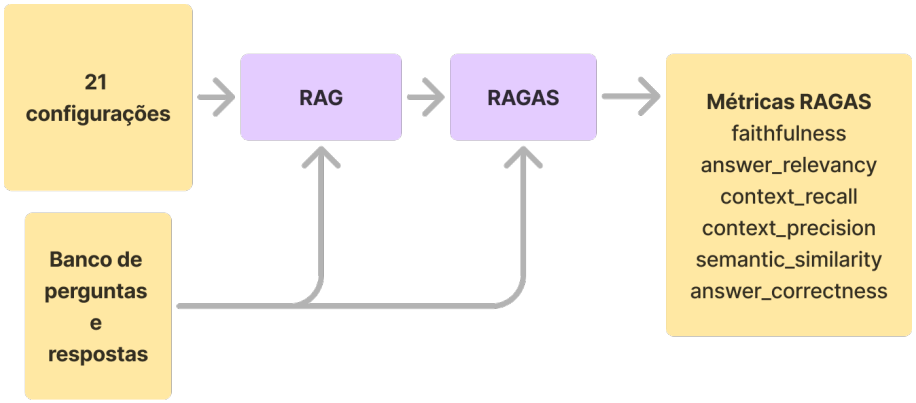

Figura 4.1: Pipeline RAG e RAGAS.

4.1 Geração de base de perguntas e respostas

O texto bruto dos documentos foi extraído utilizando a biblioteca *PyPDFLoader*. Em seguida, o modelo de linguagem GPT4 foi utilizado para gerar 200 perguntas sobre os documentos, de forma bem distribuída para garantir que os questionamentos estivessem alinhados com o todos os conteúdos da base de conhecimento. Juntamente com as perguntas, o modelo gerou as respostas baseadas nos mesmos documentos. A Figura 4.2 explica visualmente o processo de geração da base de teste.

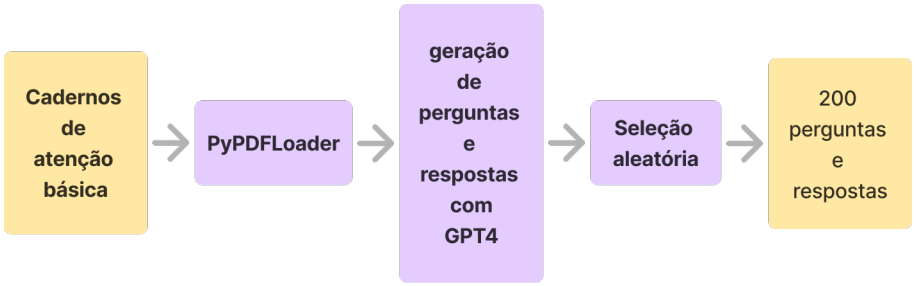

Figura 4.2: Geração de perguntas e respostas

Seguem abaixo alguns exemplos de perguntas e respostas geradas.

- Pergunta 108

  Pergunta: Com que frequência as pessoas normotensas devem aferir a pressão arterial?

  Resposta: "As pessoas que apresentarem PA entre 130/85mmHg são consideradas normotensas e deverão realizar a aferição anualmente."

- Pergunta 110

  Pergunta: Como a consulta de enfermagem deve ser abordada para os indivíduos com PA limítrofe?

  Resposta: "Consultas individuais ou coletivas para incentivar a MEV para adoção de hábitos saudáveis são recomendadas na prevenção primária da HAS, notadamente nos indivíduos com PA limítrofe."

- Pergunta 140

  Pergunta: Quais são os fatores principais que podem ser agrupados nas razões para fumar?

  Resposta: "As razões para fumar são agrupadas em nove fatores principais, sendo eles: dependência, prazer de fumar, redução da tensão, estimulação, automatismo, manuseio, tabagismo social, controle de peso e associação estreita."

4.2 Construção dos VectorStores

Foram criadas 18 *vectorStores* utilizando combinações distintas de configuradores de divisão de texto (*splitters*), tamanhos de sobreposição de fragmentos (*chunk overlap*) e métricas de similaridade (ip, l2 e cosine) com base nos documentos anteriormente selecionados. As configurações selecionadas estão discriminadas na Tabela 4.1.

Número	Splitter	Separador	Chunk Overlap	Métrica
1	RecursiveCharacterTextSplitter	`\n\n, \n`	50	ip
2	RecursiveCharacterTextSplitter	`\n\n, \n`	50	l2
3	RecursiveCharacterTextSplitter	`\n\n, \n`	50	cosine
4	RecursiveCharacterTextSplitter	`\n\n, \n`	100	ip
5	RecursiveCharacterTextSplitter	`\n\n, \n`	100	l2
6	RecursiveCharacterTextSplitter	`\n\n, \n`	100	cosine
7	RecursiveCharacterTextSplitter	`\n\n, \n, .`	50	ip
8	RecursiveCharacterTextSplitter	`\n\n, \n, .`	50	l2
9	RecursiveCharacterTextSplitter	`\n\n, \n, .`	50	cosine
10	RecursiveCharacterTextSplitter	`\n\n, \n, .`	100	ip
11	RecursiveCharacterTextSplitter	`\n\n, \n, .`	100	l2
12	RecursiveCharacterTextSplitter	`\n\n, \n, .`	100	cosine
13	CharacterTextSplitter	`\n`		ip
14	CharacterTextSplitter	`\n`		l2
15	CharacterTextSplitter	`\n`		cosine
16	CharacterTextSplitter	.		ip
17	CharacterTextSplitter	.		l2
18	CharacterTextSplitter	.		cosine

Tabela 4.1: Configurações de cada *vectorStore*

Segue abaixo uma breve explicação de cada variável acompanhada.

- ***TextSplitter*:** Define o comportamento de como a quebra de texto ocorre a partir de diferentes classes fornecidas pelo Langchain, quebrando o texto recursivamente ou de maneira bruta.
- **Separador:** Define os pontos de quebra de texto, em quais caracteres haverá a subdivisão ou finalização do trecho. No caso do *CharacterSplitter* ele percorre até encontrar o caractere. já no *RecursiveCharacterTextSplitter* há uma busca recursiva entre separadores decompondo o texto em pedaços progressivamente menores, até se adequar ao limite de tamanho de *chunk*.

- **Chunk Overlap:** Define a sobreposição dos *chunks*, medido em *tokens*. Como o *CharacterTextSplitter* ignora definições de *chunk* não há divisão de testes para esse caso.
- **Métrica de similaridade:** Função matemática utilizada para calcular a similaridade entre a pergunta e os *chunks* de contexto sendo buscados. Na Tabela 4.1, *cosine* representa similaridade de cosseno, l2 representa distância euclidiana e ip representa produto interno.

Na Tabela 4.1, observa-se a ausência de variação da configuração de *Chunk Overlap* para os *splitters* do tipo *CharacterTextSplitter*. A ausência decorre de um problema observado na biblioteca do Langchain, na qual, o parâmetro *chunkOverlap* é ignorado para esse tipo de *Splitter*.

O separador `\n\n` também não está presente para o *CharacterTextSplitter* pois os *chunks* gerados eram grandes demais, prejudicando o desempenho de forma a se tornar inutilizável.

4.3 Utilização do RAG para responder as perguntas da base

Cada uma das 200 perguntas geradas foi submetida ao sistema RAG. O processo foi repetido para cada uma das 18 configurações de *vectorstore*, resultando em um conjunto distinto de respostas para cada variação dos parâmetros de segmentação e *vectorStore*. Além da resposta, para cada pergunta, foi armazenado também o contexto buscado pelo *retriever*.

4.4 Avaliação dos resultados utilizando RAGAS

As respostas produzidas para cada configuração foram analisadas utilizando o *framework* RAGAS. Esse processo permitiu calcular métricas quantitativas sobre a qualidade das respostas, considerando aspectos como relevância, correção, completude contextual e similaridade semântica.

As métricas escolhidas foram:

1. Faithfulness
2. Answer Relevancy
3. Context Recall

4. Context Precision
5. Semantic Similarity
6. Answer Correctness

Os resultados da avaliação foram registrados em 18 bases de dados distintas, correspondendo às diferentes configurações de *vectorstore*, possibilitando comparações e análises estatísticas dos impactos das variações na configuração do sistema.

Capítulo 5

Resultados

Nesta seção, é realizada uma análise estatística e visual das métricas extraídas pelo *framework* RAGAS, utilizando histogramas e *boxplots* para observar tendências gerais e comparações entre as configurações testadas. Além disso, é investigada a distribuição dos segmentos de texto (*chunks*) gerados para garantir que não houvesse inconsistências estruturais que pudessem comprometer a recuperação da informação.

Para nortear a análise dos resultados, o estudo busca responder as seguintes questões:

- *RC1*: Um *chatbot* baseado em RAG pode alcançar um bom desempenho em respostas a perguntas relacionadas à saúde, conforme métricas específicas?
- *RC2*: Entre as diferentes configurações testadas, é possível identificar uma configuração ideal que otimize a recuperação de informações e a geração de respostas?

Os códigos utilizados para a realização da análise, bem como os dados originais se encontram no github [1].

5.1 Análise de tamanho dos Chunks

A estrutura dos *chunks* foi examinada para identificar possíveis irregularidades no tamanho e na segmentação dos documentos processados. A Tabela 5.1 apresenta o tamanho médio, mínimo e máximo dos *chunks* para cada configuração testada.

[1] `https://github.com/Anon1929/tcc_rag_saude`

Número	**Tamanho médio Chunks**	75%	Max
1	429.63	488	512
2	429.63	488	512
3	429.63	488	512
4	440.14	488	512
5	440.14	488	512
6	440.14	488	512
7	429.63	488	512
8	429.63	488	512
9	429.63	488	512
10	440.14	488	512
11	440.14	488	512
12	440.14	488	512
13	430.46	488	512
14	430.46	488	512
15	430.46	488	512
16	452.17	495	6403
17	452.17	495	6403
18	452.17	495	6403

Tabela 5.1: Tamanho dos Chunks

Não houve uma alteração muito significativa no tamanho médio dos *chunks*. Em compensação, nos *vectorStores* utilizando *CharacterTextSplitters* com separador ".", o tamanho máximo foi muito maior, passando de 6000 *tokens*. Esse comportamento é possivelmente problemático pois alguns *chunks* passam do limite de *tokens* do *embedding* utilizado, que é de 1024 *tokens*. Esse comportamento deve ser tido em vista durante a utilização desse tipo de *splitter* em futuras implementações.

Contudo, mais de 75% da amostragem ficou abaixo de 495 *tokens*, indicando que os *chunks* de tamanho anormal são exceção e, por isso, seu impacto negativo no resultado final deve ser pequeno.

5.2 Pré-tratamento dos resultados

Durante a análise, foi realizado um tratamento de certos resultados da avaliação do RAGAS. Devido a erros da biblioteca, em algumas métricas como *Faithfulness* e *Answer correctness*, certas pontuações apresentaram valores inválidos, o que indica que o modelo não conseguiu avaliar aqueles testes em específico. Isso ocorreu em uma média de 20 casos

por configuração (10% dos casos de teste). Para manter o escopo da avaliação, os valores inválidos foram substituídos pelo valor 0.

Para exemplificar, abaixo está o teste nº 198, que apresentou falha na avaliação de *Faithfulness* apesar de apresentar contexto e respostas relacionados.

Teste nº 198

Pergunta: Quais são os principais problemas de saúde relacionados à deficiência de vitamina A?

Resposta original: A deficiência de vitamina A pode levar à cegueira, carência esta que afeta milhões de crianças no mundo.

Contexto Recuperado: A deficiência crônica de vitamina A ainda é um dos mais resistentes problemas nutricionais nos países em desenvolvimento, a despeito dos sintomas não serem de', ' • Baixa na reserva de vitamina A no organismo, pode aumentar o risco de morbidade por diarreia e infecções respiratórias e mortalidade, principalmente nos grupos de risco. • Com relação aos problemas visuais, podem ocorrer: cegueira noturna, ressecamento da conjuntiva, ressecamento da córnea, lesão da córnea e cegueira irreversível.', 'inicialmente, à cegueira noturna, evoluindo até situações mais graves como a perda da visão; • participa dos processos de defesa do organismo, contribuindo para a redução da mortalidade e da morbidade por doenças infecciosas ...

Resposta do RAG: Excelente pergunta! É fundamental compreender as consequências da deficiência de vitamina A. Segundo o CADERNO DE ATENÇÃO BÁSICA, a deficiência crônica de vitamina A pode levar a uma série de problemas de saúde graves. Principalmente, estão presentes: * Risco aumentado de morbidade e mortalidade por diarreia e infecções respiratórias; * Problemas visuais, como cegueira noturna, ressecamento da conjuntiva, ressecamento da córnea, lesão da córnea e cegueira irreversível; * Participação reduzida nos processos de defesa do organismo, o que aumenta a mortalidade e morbidade por doenças infecciosas. É essencial que os agentes comunitários de saúde estejam atentos ao risco de deficiência de vitamina A e promovam estratégias de prevenção e tratamento para evitar complicações sérias.

Tais erros indicam problemas de confiabilidade e consistência no processo de avaliação da biblioteca RAGAS, o que pode influenciar os resultados das métricas.

5.3 Análise utilizando Boxplots

Os *BoxPlots* das métricas calculadas indicaram uma tendência geral de desempenho consistente entre as diferentes configurações do *RAG*, sugerindo que o modelo apresentou um bom desempenho em todos os cenários avaliados.

Os experimentos apresentaram resultados especialmente positivos nas métricas *answer relevancy*, *semantic similarity* e *context recall*, mas resultados não tão positivos em *answer correctness*, *context precision* e *faithfulness*.

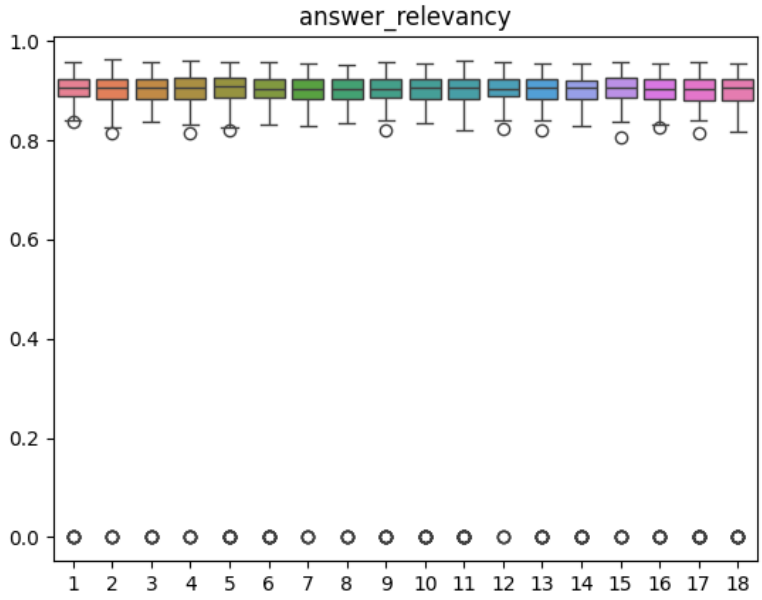

Figura 5.1: Box-plot Answer Relevancy.

A métrica **Answer Relevancy** apresentou uma média alta, próxima de 0.9, e dados pouco espalhados. O resultado indica que a resposta dada pelo modelo foi relevante para a pergunta na maior parte das vezes. É importante salientar que o fato de a resposta ser relevante não significa que a ela esteja certa.

Pelo fato da métrica levar em consideração apenas a resposta gerada e a pergunta original, ignorando o contexto recuperado, não é possível indicar se o bom desempenho ocorre devido à recuperação ou apenas ao conhecimento prévio do LLM.

A similaridade entre os resultados das diferentes configurações sugere que as variações nos parâmetros de segmentação, sobreposição de *chunks* e métricas de distância não obtiveram impacto significativo na relevância da resposta gerada em relação à pergunta original.

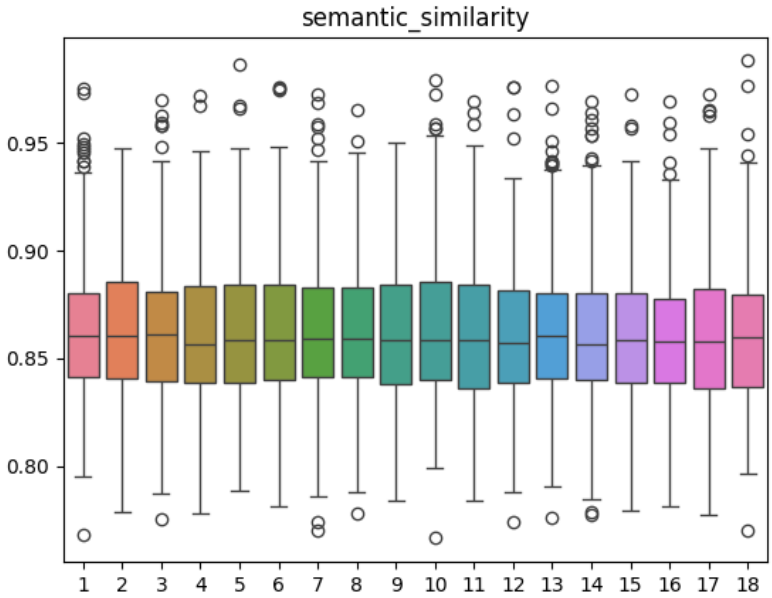

Figura 5.2: Box-plot Semantic Similarity.

A métrica **Semantic Similarity** apresentou uma média alta, próxima de 0.86, resultados pouco espalhados e ausência de *outliers* muito dispersos quando comparada a outras métricas. O resultado indica que a maioria das respostas geradas foram vetorialmente similares às respostas originais. É importante salientar que a similaridade vetorial não implica que a resposta esteja correta, mas apenas que ela aborda o mesmo assunto ou tema da resposta original..

Pelo fato da métrica levar em consideração apenas a resposta gerada e a original, ignorando o contexto recuperado, não é possível indicar se o bom desempenho ocorre devido à recuperação ou apenas ao conhecimento prévio do LLM.

Diferente da métrica *answer relevancy*, a *semantic similarity* não apresentou nenhum resultado com valor 0. Foi feita uma investigação mais a fundo para tentar elucidar esse padrão e, analisando os *outliers* de valor 0, foi possível constatar que a maioria deles se tratam do RAG respondendo que não encontrou informações sobre a pergunta na base de dados, o que demonstra que o *prompt* está bem construído e que o modelo não está gerando alucinações, fator relevante para evitar falsas informações no domínio da saúde que possam prejudicar usuários.

Nesses casos, porém, apesar de a resposta ter um baixo valor de *answer relevancy*, ela ainda traz frases e palavras que também estão presentes na resposta original, por isso a similaridade semântica continua sendo alta. Isso mostra uma das falhas graves de medir qualidade de resposta utilizando similaridade vetorial.

Na Tabela 5.2 estão elencados alguns exemplos em que o valor de *answer relevancy* foi 0 mas *semantic similarity* teve um valor alto.

semantic_similarity	user_input	resposta gerada
0.8367503396584031	O que a Lei Federal nº 9.294 proporciona em relação ao controle do tabagismo?	A Lei Federal nº 9.294 não é mencionada explicitamente no material fornecido, por isso não posso...
0.8986818346808558	O que é uma "entrevista motivacional"conforme descrito no texto?	...eu não vejo nenhuma relação entre a pergunta e o contexto fornecido...
0.8216704122800627	O que caracteriza a bulimia nervosa segundo o texto?	Não há menção à bulimia nervosa no texto fornecido. Portanto, não é possível responder...
0.8707686199357618	Quais são algumas complicações da malária grave listadas no texto?	Não há menção à malária no texto fornecido...

Tabela 5.2: Exemplos com alto semantic similarity e answer relevancy

A similaridade entre os resultados das diferentes configurações sugere que as variações nos parâmetros de segmentação, sobreposição de *chunks* e métricas de distância não obtiveram impacto significativo na similaridade entre a resposta gerada e a resposta original.

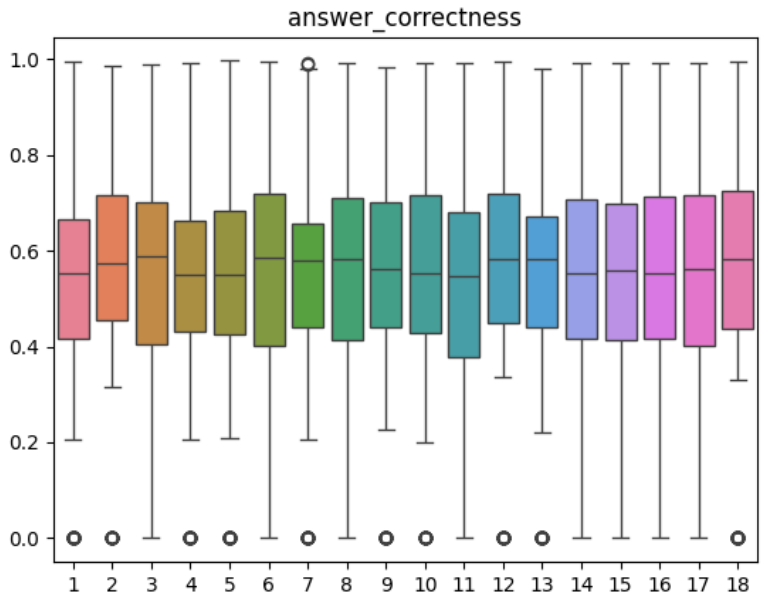

Figura 5.3: Box-plot Answer Correctness.

Answer Correctness apresentou um resultado neutro para positivo, com uma média acima de 0.5, mas dados muito espalhados. Essa métrica divide a resposta em afirmações e classifica elas em corretas ou incorretas usando uma LLM e comparando com a resposta original.

Muito provavelmente o resultado um pouco a baixo foi devido ao fato de que o *chatbot* não gera apenas a resposta da pergunta, mas sim uma mensagem amigável para ser lida por um agente comunitário de saúde. Dessa forma, é possível que esse preenchimento gerado pelo LLM seja considerado pela métrica como afirmações incorretas, abaixando a pontuação final.

A similaridade entre os resultados das diferentes configurações sugere que as variações nos parâmetros de segmentação, sobreposição de *chunks* e métricas de distância não obtiveram impacto significativo na qualidade final das respostas geradas.

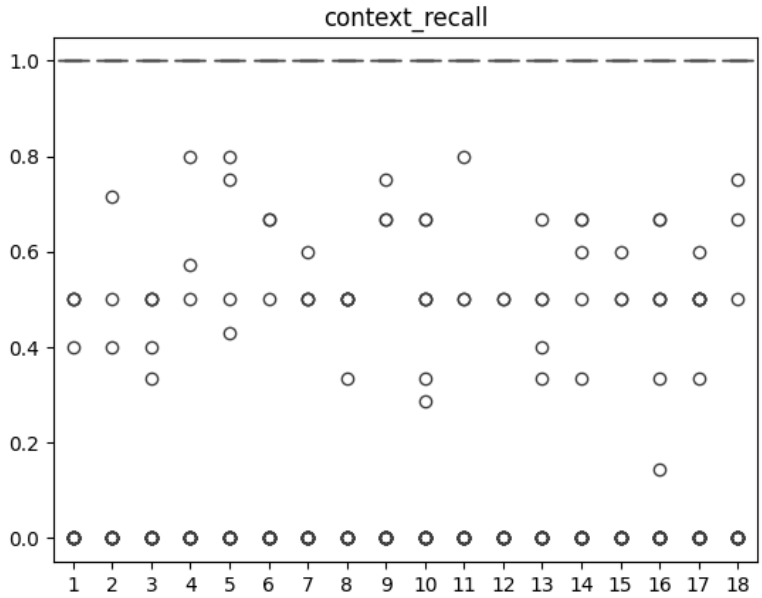

Figura 5.4: Box-plot Context Recall.

Context Recall apresentou uma média altíssima, próxima de 1, e dados extremamente concentrados próximos à média. Apresentou diversos *outliers* com resultados ruins, vários com valor 0. No entanto, a maioria dos casos de teste obtiveram pontuação perfeita na métrica.

Esse resultado significa que, cerca de 90% das vezes, todas as afirmações contidas na resposta original podiam ser inferidas de algum dos *chunks* de contexto recuperados. Isso indica que a etapa de recuperação de informações do RAG está sendo efetiva e apresenta alta confiabilidade, fator crítico para domínios sensíveis como o da saúde.

A similaridade entre os resultados das diferentes configurações sugere que as variações nos parâmetros de segmentação, sobreposição de *chunks* e métricas de distância não obtiveram impacto significativo na qualidade da recuperação de informações.

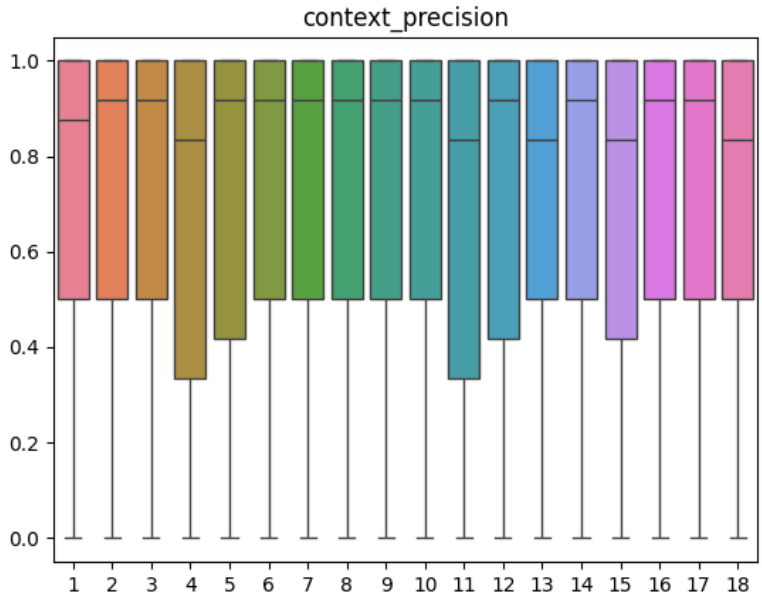

Figura 5.5: Box-plot Context Precision.

Context Precision apresentou uma média alta, próxima de 0.9, porém dados muito espalhados. Na maior parte das configurações, o terceiro quartil (75%) foi maior do que 0.5. Essa métrica mede a proporção dos *chunks* recuperados que foram relevantes para a resposta original, dessa forma, o resultado indica que possivelmente estão sendo recuperados muitos *chunks* não relevantes.

Analisando de forma conjunta o *context precision* e o resultado positivo do *context recall* é possível concluir que na maior parte das vezes é trazido um *chunk* relevante que contribui para a resposta, porém, possivelmente junto com ele são trazidos outros *chunks* que não são relevantes. Esse comportamento é preocupante pois pode gerar ruído e prejudicar o desempenho do RAG. Possíveis alterações e testes com base na quantidade de documentos recuperados podem ajudar a refinar as métricas para o resultado obtido.

A similaridade entre os resultados das diferentes configurações sugere que as variações nos parâmetros de segmentação, sobreposição de *chunks* e métricas de distância não obtiveram impacto significativo na precisão da recuperação de informações.

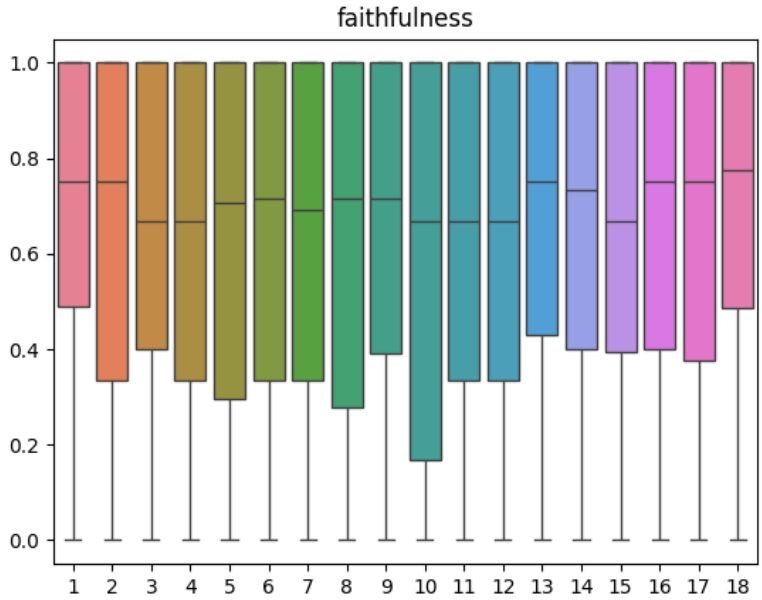

Figura 5.6: Box-plot Faifthfulness.

Faifthfulness apresentou uma média não tão alta, próxima de 0.75, e dados muito espalhados. O *Faithfulness* mede se a resposta gerada pelo RAG pode ser derivada a partir do contexto recuperado. O resultado mostra que, em vários caso, pelo menos uma parte do texto gerado pelo LLM não pôde ser derivada do contexto recuperado.

Uma possível teoria do porquê desse resultado é que, além da resposta para a pergunta, o modelo gera uma série de preenchimentos devido ao *prompt* de agente de saúde, para tornar a comunicação mais amigável e receptiva. Esse conteúdo extra talvez seja interpretado como texto que não pode ser derivado do contexto recuperado.

Além disso, houveram algumas pontuações 0. Investigando foi possível constatar que várias delas eram falsos negativos, em que a resposta estava coerente com o contexto recuperado, mas o resultado da métrica não foi condizente, conforme apresentado na seção 5.2. Um desses exemplos está apresentado a seguir:

Resposta do RAG: A longitudinalidade na Atenção Básica refere-se ao acompanhamento processual e longitudinal do cuidado à saúde das pessoas, independentemente de elas terem alguma doença. Isso significa que o trabalho na Atenção Básica é de longo prazo, envolvendo a proximidade com o usuário, seu território e sua realidade para construir um processo de atenção integral.Nesse sentido, a longitudinalidade visa ofertar à pessoa idosa e à sua rede de suporte social uma atenção humanizada, com orientação, acompanhamento e apoio domiciliar, respeitando culturas locais,

diversidades do envelhecer e reduzindo barreiras arquitetônicas. Além disso, a longitudinalidade busca produzir a atenção integral, considerando o sujeito em sua singularidade e inserção sociocultural.É fundamental que os agentes comunitários de saúde estejam atentos ao estado motivacional do sujeito e à sua realidade, analisando o que de fato está acontecendo e ajustando o acompanhamento de acordo com as necessidades específicas de cada pessoa.

Contexto recuperado:

- haverá como definir perguntas ou afirmações corretas.Já mencionamos neste capítulo que o trabalho na Atenção Básica é longitudinal, ou seja, que o cuidado à saúde das pessoas deve acontecer ao longo do tempo, independentemente do usuário estar com alguma doença. Portanto, trata-se de um acompanhamento processual. A proximidade com o usuário, seu território e sua realidade vão auxiliar a construção deste processo
- familiares, a história clínica - aspectos biológicos, psíquicos, funcionais e sociais - e o exame físico.Na Atenção Básica espera-se oferecer à pessoa idosa e à sua rede de suporte social, incluindo familiares e cuidadores (quando existente), uma atenção humanizada com orientação, acompanhamento e apoio domiciliar, com respeito às culturas locais,às diversidades do envelhecer e à diminuição das barreiras arquitetônicas de forma a
- continuidade do cuidado, da integralidade da atenção, da responsabilização, da humanização, da equidade e da participação social. A Atenção Básica considera o sujeito em sua singularidade e inserção sociocultural, buscando produzir a atenção integral.Na construção da atenção integral, a Atenção Básica deve cumprir algumas funções para contribuir com o funcionamento das Redes de Atenção à Saúde, são elas: ser base, atuando no
- da Atenção Básica dos quais a pessoa participa.É sempre importante prestar atenção ao estado motivacional do sujeito – o quanto ele está "animado" quanto a voltar a fazer as atividades, e a sua realidade –, analisando o que de fato

Durante a análise, também foi possível constatar uma quantidade significativa de resultados inválidos nas avaliações, em que o Ragas falhou em avaliar uma média de 10% dos testes. A pontuação nesses casos estava sendo substituída por 0.

Removendo esses testes inválidos, é possível observar um aumento da pontuação e uma diminuição na variância, tornando o resultado mais condizente com as outras métricas, como é possível observar na Figura 5.7.

A similaridade entre os resultados das diferentes configurações sugere que as variações nos parâmetros de segmentação, sobreposição de *chunks* e métricas de distância não obtiveram impacto significativo na fidelidade da resposta em relação ao contexto recuperado.

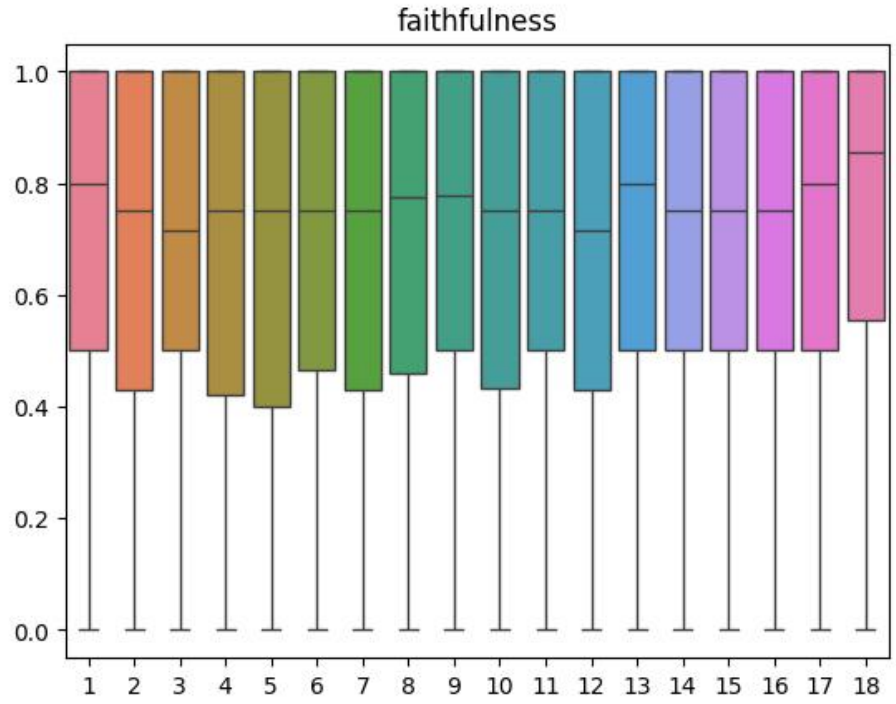

Figura 5.7: Box-plot faithfulness removendo testes inválidos

5.4 Testes Estatísticos

Para verificar a existência de diferenças estatisticamente significativas entre os resultados das diferentes configurações testadas, são realizados dois tipos de testes estatísticos:

- **Teste de Kruskal-Wallis:** aplicado a cada métrica para avaliar diferenças entre os grupos sem pressupor normalidade nos dados.
- **Teste de Levene seguido de ANOVA tradicional e de Welch:** Primeiramente testando se as variâncias são similares utilizando Levene e depois ANOVA tradicional ou de Welch para testar se havia diferença significativa entre os grupos.

5.4.1 Teste de Kruskal-Walis

A Tabela 5.3 mostra que todos os p-valores são significativamente maiores que 0,05, indicando a ausência de diferenças estatisticamente significativas entre os grupos e evidenciando a similaridade dos dados.

Métrica	H	P
faithfulness	13.6140	0.6942
answer relevancy	5.2850	0.9968
context recall	7.9702	0.9672
context precision	4.3114	0.9991
semantic similarity	2.6399	1.0000
answer correctness	5.8437	0.9942

Tabela 5.3: Teste de Kruskal-Walis

5.4.2 Teste de Levene

Como observado na Tabela 5.4, todas as métricas foram classificadas como com variância homogênea, com exceção da *answer relevancy*. Para as que possuem variância homogênea a Anova tradicional foi utilizada e para as que possuem variância heterogênea foi utilizada a Anova de Welch.

Métrica	Levene	P-valor	Homogeneidade
faithfulness	1.2796	0.1952	homogênea
answer relevancy	1.8666	0.0166	não homogênea
context recall	0.4101	0.9838	homogênea
context precision	0.4685	0.9671	homogênea
semantic similarity	0.1588	1.000	homogênea
answer correctness	0.3712	0.9908	homogênea

Tabela 5.4: Teste de Levene

5.5 Teste Anova simples e Anova de Welch

A Tabela 5.5 mostra que todos os p-valores são significativamente maiores que 0,05, indicando a ausência de diferenças estatisticamente significativas entre os grupos e evidenciando a similaridade dos dados. O mesmo pode ser observado na Tabela 5.6.

Métrica	F	P-valor
faithfulness	0.9113	0.5601
context recall	0.4101	0.9838
context precision	0.3768	0.9900
semantic similarity	0.1234	1.0000
answer correctness	0.4578	0.9708

Tabela 5.5: Anova tradicional

Métrica	F	P-valor
answer relevancy	0.0558	1.0000

Tabela 5.6: Anova de Welch

Os resultados indicaram que não há diferenças estatisticamente significativas entre as configurações testadas, indicando que as variações na segmentação e no método de indexação não impactaram de forma relevante o desempenho do modelo.

Capítulo 6

Conclusão, Riscos à validade e Trabalhos futuros

6.1 Conclusão

Todos os objetivos definidos para o trabalho foram cumpridos. Foi implementado um *chatbot* para respostas de perguntas médicas utilizando RAG, a ferramenta foi disponibilizada por meio de um bot na ferramenta de mensagens Telegram, foi criada uma base de testes de forma automatizada e o desempenho do sistema foi avaliado utilizando as métricas do Ragas.

Com base nos resultados obtidos nos experimentos, é possível responder à *RC1*, dado que, dentre as métricas observadas, todas, exceto *answer correctness*, apresentaram boas médias e pontuações em diferentes contextos de avaliação. No entanto, há espaço para aprimoramentos, especialmente no que se refere à avaliação da usabilidade prática do chatbot e à realização de testes manuais que possam capturar nuances não detectadas pela análise automatizada.

Ademais, referente à *RC2*, não foi possível identificar uma configuração claramente superior dentre as analisadas, uma vez que todas as combinações de parâmetros apresentaram resultados estatisticamente equivalentes nos testes de Kruskal e Anova.

Além disso, o estudo reforça que a avaliação de modelos de linguagem e sistemas RAG ainda é um campo em evolução, com desafios metodológicos significativos. A dependência de métricas automatizadas, como as fornecidas pelo *framework* RAGAS, pode introduzir limitações na interpretação dos resultados, uma vez que essas métricas estão sujeitas às limitações dos próprios modelos de linguagem utilizados na avaliação.

6.2 Riscos à validade

O principal risco à validade dos resultados está na forma de metrificação empregada. Embora o RAGAS represente um avanço na avaliação automatizada de sistemas RAG, sua eficácia ainda é condicionada à qualidade e limitações do modelo utilizado para calcular as métricas. Ademais, a observabilidade das métricas do RAGAS, pela natureza de como são construídas e calculadas é precária. O *framework* é abstrai seu funcionamento interno, o que dificulta a traçar pontos de falha exatos e a entender por completo os resultados. Futuras versões da biblioteca possivelmente remediarão esse fator, visto que atualizações mais recentes melhoraram o sistema de *logging*.

Além disso, a base de perguntas e respostas foi gerada automaticamente utilizando um modelo de linguagem, o que pode gerar vieses e representa um cenário de uso muito diferente do real.

É importante salientar também, que o *framework* Ragas atualizou e mudou suas *APIs* diversas vezes durante o desenvolvimento deste trabalho, dificultando a implementação e aumentando o risco de possíveis erros no cálculo das métricas, sendo possível exemplificar a alteração da métrica *Answer Correctness* em versões mais recentes, o que comprometeu a análise em alguns aspectos devido a dificuldades com a documentação. O mesmo comportamento foi observado com o *Langchain*, porém em uma escala bem menor.

Em contraste, a avaliação humana, apesar de ser mais custosa e subjetiva, tende a ser mais robusta, uma vez que pode lidar melhor com nuances de contexto e interpretação que os modelos automatizados podem não captar.

6.3 Trabalhos Futuros

Para aprofundar os achados deste estudo, sugere-se a realização de novos experimentos que explorem configurações não abordadas, incluindo ajustes mais extremos dos parâmetros de segmentação e novas técnicas para otimizar o desempenho do RAG. Além disso, recomenda-se a incorporação da avaliação humana como um complemento à análise automatizada, permitindo uma triagem mais precisa da qualidade das respostas e da usabilidade do *chatbot* em um ambiente real.

Este trabalho apresentou uma abordagem baseada no modelo Llama3 e na utilização de um *vectorStore* para recuperação de informações na área da saúde. No entanto, diversas outras possibilidades podem ser exploradas para aprimorar o desempenho e a eficiência do sistema.

Uma das direções futuras é a avaliação de outros modelos de linguagem além do Llama3, a fim de identificar alternativas que com melhor desempenho, precisão e contex-

tualização das respostas. Além disso, outras estratégias de recuperação de informações podem ser exploradas além do uso de *vectorStore*, por exemplo, o uso de grafo de conhecimento.

Outra linha envolve a segmentação de texto. Poderiam ser testadas diferentes formas de dividir os documentos, considerando não apenas caracteres separadores, mas também a estrutura do conteúdo. Outra possibilidade é a utilização de modelos de linguagem para separação semântica dos textos. Além disso, técnicas de pré-processamento poderiam ser aplicadas para remover partes irrelevantes ou redundantes do contexto, tornando a recuperação mais eficiente.

Por fim, a avaliação dos resultados pode ser aprimorada com a incorporação da avaliação humana como um complemento à análise automatizada, permitindo uma triagem mais precisa da qualidade das respostas e da usabilidade do *chatbot* em um ambiente real. Também podem ser utilizados outros *frameworks* automatizados específicos de avaliação para sistemas RAG. Essas estratégias podem fornecer uma análise mais completa e confiável do desempenho do sistema.

Os pontos destacados abrem espaço para futuras pesquisas e experimentações, permitindo avanços na implementação e na eficácia de sistemas baseados em RAG para aplicações na área da saúde.

Referências

[Ba et al., 2016] Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. 11

[Bahdanau et al., 2016] Bahdanau, D., Cho, K., and Bengio, Y. (2016). Neural machine translation by jointly learning to align and translate. 11

[Commons, 2024] Commons, W. (2024). Rag diagram - wikimedia commons cc by. Acessado em: 12 de fevereiro de 2025, disponível em https://commons.m.wikimedia.org/wiki/File:RAG$_d$*iagram.svg.ix*, 14

[Conneau et al., 2020] Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., and Stoyanov, V. (2020). Unsupervised cross-lingual representation learning at scale. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J., editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics. 8

[Devlin et al., 2019] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In Burstein, J., Doran, C., and Solorio, T., editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics. 7, 9

[dvgodoy, 2025] dvgodoy (2025). Repositório de imagens cc by. Acessado em: 12 de fevereiro de 2025 disponível em https://github.com/dvgodoy/dl-visuals?tab=readme-ov-file. ix, 10, 12, 13

[Es et al., 2024] Es, S., James, J., Espinosa Anke, L., and Schockaert, S. (2024). RAGAs: Automated evaluation of retrieval augmented generation. In Aletras, N. and De Clercq, O., editors, *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158, St. Julians, Malta. Association for Computational Linguistics. 20

[Gams et al., 2024] Gams, M., Smerkol, M., Kocuvan, P., and Zadobovšek, M. (2024). *Developing a Medical Chatbot: Integrating Medical Knowledge into GPT for Healthcare Applications.* 2

[Grattafiori et al., 2024] Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., Yang, A., Fan, A.,

Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Roziere, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C. C., Nikolaidis, C., Allonsius, D., Song, D., Pintz, D., Livshits, D., Wyatt, D., Esiobu, D., Choudhary, D., Mahajan, D., Garcia-Olano, D., Perino, D., Hupkes, D., Lakomkin, E., AlBadawy, E., Lobanova, E., Dinan, E., Smith, E. M., Radenovic, F., Guzmán, F., Zhang, F., Synnaeve, G., Lee, G., Anderson, G. L., Thattai, G., Nail, G., Mialon, G., Pang, G., Cucurell, G., Nguyen, H., Korevaar, H., Xu, H., Touvron, H., Zarov, I., Ibarra, I. A., Kloumann, I., Misra, I., Evtimov, I., Zhang, J., Copet, J., Lee, J., Geffert, J., Vranes, J., Park, J., Mahadeokar, J., Shah, J., van der Linde, J., Billock, J., Hong, J., Lee, J., Fu, J., Chi, J., Huang, J., Liu, J., Wang, J., Yu, J., Bitton, J., Spisak, J., Park, J., Rocca, J., Johnstun, J., Saxe, J., Jia, J., Alwala, K. V., Prasad, K., Upasani, K., Plawiak, K., Li, K., Heafield, K., Stone, K., El-Arini, K., Iyer, K., Malik, K., Chiu, K., Bhalla, K., Lakhotia, K., Rantala-Yeary, L., van der Maaten, L., Chen, L., Tan, L., Jenkins, L., Martin, L., Madaan, L., Malo, L., Blecher, L., Landzaat, L., de Oliveira, L., Muzzi, M., Pasupuleti, M., Singh, M., Paluri, M., Kardas, M., Tsimpoukelli, M., Oldham, M., Rita, M., Pavlova, M., Kambadur, M., Lewis, M., Si, M., Singh, M. K., Hassan, M., Goyal, N., Torabi, N., Bashlykov, N., Bogoychev, N., Chatterji, N., Zhang, N., Duchenne, O., Çelebi, O., Alrassy, P., Zhang, P., Li, P., Vasic, P., Weng, P., Bhargava, P., Dubal, P., Krishnan, P., Koura, P. S., Xu, P., He, Q., Dong, Q., Srinivasan, R., Ganapathy, R., Calderer, R., Cabral, R. S., Stojnic, R., Raileanu, R., Maheswari, R., Girdhar, R., Patel, R., Sauvestre, R., Polidoro, R., Sumbaly, R., Taylor, R., Silva, R., Hou, R., Wang, R., Hosseini, S., Chennabasappa, S., Singh, S., Bell, S., Kim, S. S., Edunov, S., Nie, S., Narang, S., Raparthy, S., Shen, S., Wan, S., Bhosale, S., Zhang, S., Vandenhende, S., Batra, S., Whitman, S., Sootla, S., Collot, S., Gururangan, S., Borodinsky, S., Herman, T., Fowler, T., Sheasha, T., Georgiou, T., Scialom, T., Speckbacher, T., Mihaylov, T., Xiao, T., Karn, U., Goswami, V., Gupta, V., Ramanathan, V., Kerkez, V., Gonguet, V., Do, V., Vogeti, V., Albiero, V., Petrovic, V., Chu, W., Xiong, W., Fu, W., Meers, W., Martinet, X., Wang, X., Wang, X., Tan, X. E., Xia, X., Xie, X., Jia, X., Wang, X., Goldschlag, Y., Gaur, Y., Babaei, Y., Wen, Y., Song, Y., Zhang, Y., Li, Y., Mao, Y., Coudert, Z. D., Yan, Z., Chen, Z., Papakipos, Z., Singh, A., Srivastava, A., Jain, A., Kelsey, A., Shajnfeld, A., Gangidi, A., Victoria, A., Goldstand, A., Menon, A., Sharma, A., Boesenberg, A., Baevski, A., Feinstein, A., Kallet, A., Sangani, A., Teo, A., Yunus, A., Lupu, A., Alvarado, A., Caples, A., Gu, A., Ho, A., Poulton, A., Ryan, A., Ramchandani, A., Dong, A., Franco, A., Goyal, A., Saraf, A., Chowdhury, A., Gabriel, A., Bharambe, A., Eisenman, A., Yazdan, A., James, B., Maurer, B., Leonhardi, B., Huang, B., Loyd, B., Paola, B. D., Paranjape, B., Liu, B., Wu, B., Ni, B., Hancock, B., Wasti, B., Spence, B., Stojkovic, B., Gamido, B., Montalvo, B., Parker, C., Burton, C., Mejia, C., Liu, C., Wang, C., Kim, C., Zhou, C., Hu, C., Chu, C.-H., Cai, C., Tindal, C., Feichtenhofer, C., Gao, C., Civin, D., Beaty, D., Kreymer, D., Li, D., Adkins, D., Xu, D., Testuggine, D., David, D., Parikh, D., Liskovich, D., Foss, D., Wang, D., Le, D., Holland, D., Dowling, E., Jamil, E., Montgomery, E., Presani, E., Hahn, E., Wood, E., Le, E.-T., Brinkman, E., Arcaute, E., Dunbar, E., Smothers, E., Sun, F., Kreuk, F., Tian, F., Kokkinos, F., Ozgenel, F., Caggioni, F., Kanayet, F., Seide, F., Florez, G. M., Schwarz, G., Badeer, G., Swee, G., Halpern, G., Herman, G., Sizov, G., Guangyi, Zhang, Lakshminarayanan, G., Inan, H., Shojanazeri, H., Zou, H., Wang,

H., Zha, H., Habeeb, H., Rudolph, H., Suk, H., Aspegren, H., Goldman, H., Zhan, H., Damlaj, I., Molybog, I., Tufanov, I., Leontiadis, I., Veliche, I.-E., Gat, I., Weissman, J., Geboski, J., Kohli, J., Lam, J., Asher, J., Gaya, J.-B., Marcus, J., Tang, J., Chan, J., Zhen, J., Reizenstein, J., Teboul, J., Zhong, J., Jin, J., Yang, J., Cummings, J., Carvill, J., Shepard, J., McPhie, J., Torres, J., Ginsburg, J., Wang, J., Wu, K., U, K. H., Saxena, K., Khandelwal, K., Zand, K., Matosich, K., Veeraraghavan, K., Michelena, K., Li, K., Jagadeesh, K., Huang, K., Chawla, K., Huang, K., Chen, L., Garg, L., A, L., Silva, L., Bell, L., Zhang, L., Guo, L., Yu, L., Moshkovich, L., Wehrstedt, L., Khabsa, M., Avalani, M., Bhatt, M., Mankus, M., Hasson, M., Lennie, M., Reso, M., Groshev, M., Naumov, M., Lathi, M., Keneally, M., Liu, M., Seltzer, M. L., Valko, M., Restrepo, M., Patel, M., Vyatskov, M., Samvelyan, M., Clark, M., Macey, M., Wang, M., Hermoso, M. J., Metanat, M., Rastegari, M., Bansal, M., Santhanam, N., Parks, N., White, N., Bawa, N., Singhal, N., Egebo, N., Usunier, N., Mehta, N., Laptev, N. P., Dong, N., Cheng, N., Chernoguz, O., Hart, O., Salpekar, O., Kalinli, O., Kent, P., Parekh, P., Saab, P., Balaji, P., Rittner, P., Bontrager, P., Roux, P., Dollar, P., Zvyagina, P., Ratanchandani, P., Yuvraj, P., Liang, Q., Alao, R., Rodriguez, R., Ayub, R., Murthy, R., Nayani, R., Mitra, R., Parthasarathy, R., Li, R., Hogan, R., Battey, R., Wang, R., Howes, R., Rinott, R., Mehta, S., Siby, S., Bondu, S. J., Datta, S., Chugh, S., Hunt, S., Dhillon, S., Sidorov, S., Pan, S., Mahajan, S., Verma, S., Yamamoto, S., Ramaswamy, S., Lindsay, S., Lindsay, S., Feng, S., Lin, S., Zha, S. C., Patil, S., Shankar, S., Zhang, S., Zhang, S., Wang, S., Agarwal, S., Sajuyigbe, S., Chintala, S., Max, S., Chen, S., Kehoe, S., Satterfield, S., Govindaprasad, S., Gupta, S., Deng, S., Cho, S., Virk, S., Subramanian, S., Choudhury, S., Goldman, S., Remez, T., Glaser, T., Best, T., Koehler, T., Robinson, T., Li, T., Zhang, T., Matthews, T., Chou, T., Shaked, T., Vontimitta, V., Ajayi, V., Montanez, V., Mohan, V., Kumar, V. S., Mangla, V., Ionescu, V., Poenaru, V., Mihailescu, V. T., Ivanov, V., Li, W., Wang, W., Jiang, W., Bouaziz, W., Constable, W., Tang, X., Wu, X., Wang, X., Wu, X., Gao, X., Kleinman, Y., Chen, Y., Hu, Y., Jia, Y., Qi, Y., Li, Y., Zhang, Y., Zhang, Y., Adi, Y., Nam, Y., Yu, Wang, Zhao, Y., Hao, Y., Qian, Y., Li, Y., He, Y., Rait, Z., DeVito, Z., Rosnbrick, Z., Wen, Z., Yang, Z., Zhao, Z., and Ma, Z. (2024). The llama 3 herd of models. 17

[Guo et al., 2020] Guo, R., Sun, P., Lindgren, E., Geng, Q., Simcha, D., Chern, F., and Kumar, S. (2020). Accelerating large-scale inference with anisotropic vector quantization. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org. 15

[Han et al., 2025] Han, H., Wang, Y., Shomer, H., Guo, K., Ding, J., Lei, Y., Halappanavar, M., Rossi, R. A., Mukherjee, S., Tang, X., He, Q., Hua, Z., Long, B., Zhao, T., Shah, N., Javari, A., Xia, Y., and Tang, J. (2025). Retrieval-augmented generation with graphs (graphrag). 16

[IBGE,] IBGE. IBGE | Biblioteca | Detalhes | Pessoas com deficiência : 2022 / IBGE, Coordenação de Pesquisas por Amostra de Domicílios, disponível em https://biblioteca.ibge.gov.br/index.php/biblioteca-catalogo?view=detalhesid=2102013. 1

[Johnson et al., 2021] Johnson, J., Douze, M., and Jégou, H. (2021). Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547. 15

[Jurafsky and Martin, 2025] Jurafsky, D. and Martin, J. H. (2025). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. Prentice Hall PTR, 3rd edition. Online manuscript released January 12, 2025. 5

[Kim and Min, 2024] Kim, J. and Min, M. (2024). From rag to qa-rag: Integrating generative ai for pharmaceutical regulatory compliance process. 2

[Lewis et al., 2020] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., and Kiela, D. (2020). 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. Curran Associates Inc. 13, 14

[Lima et al., 2024] Lima, G., Macedo, C., Costa, B., and Soler, O. (2024). Acesso e qualidade de bases de dados e sistemas de informações em saúde no brasil: Revisão de escopo. *Research, Society and Development*, 13:e8413445603. 2

[Lin, 2004] Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, Barcelona, Spain. Association for Computational Linguistics. 18

[Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. 6

[Mota and Bousquat, 2023] Mota, P. H. d. S. and Bousquat, A. (2023). Desafios para a implementação da rede de cuidados à pessoa com deficiência em uma região de saúde: um olhar a partir das dimensões política, organização e estrutura. *Saúde e Sociedade*, 32(2):e220608pt. 1

[OpenAI et al., 2024] OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., Belgum, J., Bello, I., Berdine, J., Bernadett-Shapiro, G., Berner, C., Bogdonoff, L., Boiko, O., Boyd, M., Brakman, A.-L., Brockman, G., Brooks, T., Brundage, M., Button, K., Cai, T., Campbell, R., Cann, A., Carey, B., Carlson, C., Carmichael, R., Chan, B., Chang, C., Chantzis, F., Chen, D., Chen, S., Chen, R., Chen, J., Chen, M., Chess, B., Cho, C., Chu, C., Chung, H. W., Cummings, D., Currier, J., Dai, Y., Decareaux, C., Degry, T., Deutsch, N., Deville, D., Dhar, A., Dohan, D., Dowling, S., Dunning, S., Ecoffet, A., Eleti, A., Eloundou, T., Farhi, D., Fedus, L., Felix, N., Fishman, S. P., Forte, J., Fulford, I., Gao, L., Georges, E., Gibson, C., Goel, V., Gogineni, T., Goh, G., Gontijo-Lopes, R., Gordon, J., Grafstein, M., Gray, S., Greene, R., Gross, J., Gu, S. S., Guo, Y., Hallacy, C., Han, J., Harris, J., He, Y., Heaton, M., Heidecke, J., Hesse, C., Hickey, A., Hickey, W., Hoeschele, P., Houghton, B., Hsu, K., Hu, S., Hu, X., Huizinga, J., Jain, S., Jain, S., Jang, J., Jiang, A., Jiang, R., Jin, H., Jin, D., Jomoto, S., Jonn, B., Jun, H., Kaftan, T., Łukasz Kaiser, Kamali, A., Kanitscheider, I., Keskar, N. S., Khan, T., Kilpatrick, L., Kim, J. W., Kim, C., Kim, Y., Kirchner, J. H., Kiros, J., Knight, M., Kokotajlo, D., Łukasz Kondraciuk, Kondrich, A., Konstantinidis, A., Kosic, K., Krueger, G., Kuo, V., Lampe, M., Lan, I., Lee, T., Leike, J., Leung, J., Levy, D., Li, C. M., Lim, R., Lin, M., Lin, S., Litwin,

M., Lopez, T., Lowe, R., Lue, P., Makanju, A., Malfacini, K., Manning, S., Markov, T., Markovski, Y., Martin, B., Mayer, K., Mayne, A., McGrew, B., McKinney, S. M., McLeavey, C., McMillan, P., McNeil, J., Medina, D., Mehta, A., Menick, J., Metz, L., Mishchenko, A., Mishkin, P., Monaco, V., Morikawa, E., Mossing, D., Mu, T., Murati, M., Murk, O., Mély, D., Nair, A., Nakano, R., Nayak, R., Neelakantan, A., Ngo, R., Noh, H., Ouyang, L., O'Keefe, C., Pachocki, J., Paino, A., Palermo, J., Pantuliano, A., Parascandolo, G., Parish, J., Parparita, E., Passos, A., Pavlov, M., Peng, A., Perelman, A., de Avila Belbute Peres, F., Petrov, M., de Oliveira Pinto, H. P., Michael, Pokorny, Pokrass, M., Pong, V. H., Powell, T., Power, A., Power, B., Proehl, E., Puri, R., Radford, A., Rae, J., Ramesh, A., Raymond, C., Real, F., Rimbach, K., Ross, C., Rotsted, B., Roussez, H., Ryder, N., Saltarelli, M., Sanders, T., Santurkar, S., Sastry, G., Schmidt, H., Schnurr, D., Schulman, J., Selsam, D., Sheppard, K., Sherbakov, T., Shieh, J., Shoker, S., Shyam, P., Sidor, S., Sigler, E., Simens, M., Sitkin, J., Slama, K., Sohl, I., Sokolowsky, B., Song, Y., Staudacher, N., Such, F. P., Summers, N., Sutskever, I., Tang, J., Tezak, N., Thompson, M. B., Tillet, P., Tootoonchian, A., Tseng, E., Tuggle, P., Turley, N., Tworek, J., Uribe, J. F. C., Vallone, A., Vijayvergiya, A., Voss, C., Wainwright, C., Wang, J. J., Wang, A., Wang, B., Ward, J., Wei, J., Weinmann, C., Welihinda, A., Welinder, P., Weng, J., Weng, L., Wiethoff, M., Willner, D., Winter, C., Wolrich, S., Wong, H., Workman, L., Wu, S., Wu, J., Wu, M., Xiao, K., Xu, T., Yoo, S., Yu, K., Yuan, Q., Zaremba, W., Zellers, R., Zhang, C., Zhang, M., Zhao, S., Zheng, T., Zhuang, J., Zhuk, W., and Zoph, B. (2024). Gpt-4 technical report. 9, 17

[Papineni et al., 2002] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA. Association for Computational Linguistics. 18

[Peng et al., 2024] Peng, B., Zhu, Y., Liu, Y., Bo, X., Shi, H., Hong, C., Zhang, Y., and Tang, S. (2024). Graph retrieval-augmented generation: A survey. 16

[Raffel et al., 2020] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1). 17

[Ragas, 2025] Ragas (2025). Ragas documentation. Acessado em: 10 de fevereiro de 2025. 20

[Robertson and Zaragoza, 2009] Robertson, S. and Zaragoza, H. (2009). The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389. 15

[Roychowdhury et al., 2024] Roychowdhury, S., Soman, S., Ranjani, H. G., Gunda, N., Chhabra, V., and Bala, S. K. (2024). Evaluation of rag metrics for question answering in the telecom domain. 20

[Sammut and Webb, 2017] Sammut, C. and Webb, G. I. (2017). *Encyclopedia of Machine Learning and Data Mining*. Springer Publishing Company, Incorporated, 2nd edition. 6

[Sazli, 2006] Sazli, M. (2006). A brief review of feed-forward neural networks. *Communications Faculty Of Science University of Ankara*, 50:11–17. 13

[Touvron et al., 2023] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. (2023). Llama: Open and efficient foundation language models. 17

[TruEra, 2025] TruEra (2025). Trulens. Acessado em: 10 de fevereiro de 2025. 19

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc. 8, 9, 11

[Wang et al., 2024] Wang, L., Yang, N., Huang, X., Yang, L., Majumder, R., and Wei, F. (2024). Multilingual e5 text embeddings: A technical report. 8

[Yu et al., 2024a] Yu, H., Gan, A., Zhang, K., Tong, S., Liu, Q., and Liu, Z. (2024a). Evaluation of retrieval-augmented generation: A survey. 2

[Yu et al., 2024b] Yu, X., Cheng, H., Liu, X., Roth, D., and Gao, J. (2024b). Reeval: Automatic hallucination evaluation for retrieval-augmented large language models via transferable adversarial attacks. 19

[Zhang et al., 2020] Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., and Artzi, Y. (2020). Bertscore: Evaluating text generation with bert. 19

[Zhao et al., 2024] Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., Liu, P., Nie, J.-Y., and Wen, J.-R. (2024). A survey of large language models. 8

[Zhuang et al., 2021] Zhuang, L., Wayne, L., Ya, S., and Jun, Z. (2021). A robustly optimized BERT pre-training approach with post-training. In Li, S., Sun, M., Liu, Y., Wu, H., Liu, K., Che, W., He, S., and Rao, G., editors, *Proceedings of the 20th Chinese National Conference on Computational Linguistics*, pages 1218–1227, Huhhot, China. Chinese Information Processing Society of China. 8