Capítulo 6

Respostas a perguntas de clínica médica utilizando a Geração Aumentada por Recuperação (RAG)

Luciana Bencke

Abstract

Large Language Models (LLMs) have excelled in clinical tasks, but they still face important challenges, such as generating factually incorrect responses (hallucinations) and the limitation of not reflecting recent updates in medical guidelines. These risks become critical in high-impact contexts, such as answering clinical questions. Retrieval Augmented Generation (RAG) seeks to mitigate these flaws by combining the capabilities of LLMs with information retrieved from external sources, public or private, enabling more accurate, up-to-date responses that are aligned with current clinical knowledge. Applications in healthcare include virtual assistants, decision support, educational systems, and other intelligent tools. This chapter discusses the motivation for using RAG in healthcare, its components, pipeline stages, and research opportunities in the area.

Resumo

Grandes Modelos de Linguagem (LLMs) têm se destacado em tarefas clínicas, mas ainda enfrentam desafios importantes, como a geração de respostas factualmente incorretas (alucinações) e a limitação de não refletirem atualizações recentes em diretrizes médicas. Esses riscos tornam-se críticos em contextos de alto impacto, como respostas a perguntas de clínica médica. A Geração Aumentada por Recuperação (RAG) busca mitigar essas falhas ao combinar a capacidade dos LLMs com informações recuperadas de fontes externas, públicas ou privadas, possibilitando respostas mais precisas, atualizadas e alinhadas com o conhecimento clínico vigente. Aplicações em saúde incluem assistentes virtuais, suporte à decisão, sistemas educacionais e outras ferramentas inteligentes. Esse capítulo aborda a motivação para o uso do RAG na saúde, seus componentes, etapas do pipeline e oportunidades de pesquisa na área.

6.1. Introdução

Sistemas de resposta a perguntas são criados com o objetivo de suprir as demandas por informação dos seres humanos. Como grande parte do conhecimento está registrada em textos na internet, em e-mails ou em livros, esses sistemas têm forte relação com os buscadores. Atualmente, a distinção entre sistemas de perguntas e respostas e buscadores está cada vez mais tênue, à medida que os mecanismos de busca modernos incorporam grandes modelos de linguagem de grande especificamente treinados para responder a esse tipo de solicitação [Jurafsky and Martin 2025].

O escopo de um sistema de perguntas e respostas (*Question Answering*, QA) na área da saúde pode ser amplo, cobrindo uma diversidade de temas médicos e biomédicos, ou restrito, focando em subdomínios específicos, como cardiologia, oncologia ou saúde pública. A base de conhecimento utilizada por esses sistemas pode assumir múltiplas formas: desde documentos não estruturados, como prontuários, artigos científicos e diretrizes clínicas, até fontes estruturadas, como bancos de dados de saúde, ou ainda conhecimento embutido nos próprios parâmetros de modelos de linguagem, a chamada memória paramétrica.

As perguntas feitas a sistemas de QA também variam em complexidade e finalidade, sendo classificadas em factuais e não factuais [Cortes et al. 2024]. Perguntas factuais requerem respostas diretas e objetivas baseadas em evidências específicas, por exemplo "*Qual é a dosagem recomendada de paracetamol para adultos?*"ou "*Quais são os sintomas iniciais da COVID-19?*". Já perguntas não factuais exigem respostas mais extensas e contextualizadas, que envolvem a síntese de múltiplas fontes e maior elaboração interpretativa, por exemplo "*Explique as diferenças entre os protocolos de tratamento para hipertensão em idosos e adultos jovens*". Em ambientes clínicos, responder corretamente a ambos os tipos de perguntas é fundamental para garantir segurança, precisão e suporte à tomada de decisão baseada em evidências.

Aplicações baseadas em grandes modelos de linguagem (*Large Language Models*, LLMs), como o ChatGPT, exibem grande potencial em responder perguntas, mas apresentam vulnerabilidades como alucinações, vieses e imprecisões factuais — limitações especialmente críticas em ambientes clínicos, onde a exatidão das informações é essencial [Zakka et al. 2024]. LLMs têm o potencial de transformar a distribuição de informações médicas, destacando-se na geração de conteúdo e na comunicação interativa. No entanto, enfrentam limitações como desatualização, alucinações factuais e dependência de dados públicos, o que restringe seu uso na saúde [Ng et al. 2025].

Embora LLMs tenham alcançado excelente desempenho em várias tarefas que exigem resposta a perguntas de clínica médica, eles ainda enfrentam desafios com alucinações e conhecimento desatualizado [Xiong et al. 2024]. Os LLMs podem gerar respostas que parecem plausíveis, mas factualmente incorretas, conhecidas como alucinação [Chowdhury et al. 2025]. Além disso, os corpora de treinamento dos LLMs podem não incluir o conhecimento mais recente, como atualizações de diretrizes clínicas. Essas questões podem ser perigosas em domínios de alto risco, como assistência médica [Xiong et al. 2024]. [Singhal et al. 2023] avaliaram LLMs em respostas abertas sobre conhecimento clínico ao longo de eixos como factualidade, compreensão, raciocínio, possível dano e viés. Os autores constatam limitações dos modelos da época, reforçando a

importância das estruturas de avaliação e do desenvolvimento de métodos na criação de LLMs seguros e úteis para aplicações clínicas.

A Recuperação Aumentada por Geração (*Retrieval-Augmented Generation*, RAG) surge como uma solução para minimizar esses riscos na geração de respostas a perguntas clínicas, conectando LLMs a fontes externas — como artigos científicos, compêndios médicos e políticas institucionais — e permitindo acesso a informações além dos dados de treinamento. Com isso, ferramentas de Inteligência Artificial (IA) generativa podem integrar dados públicos e privados, ampliando sua aplicabilidade e precisão em contextos clínicos [Ng et al. 2025].

RAG combina técnicas de recuperação de informações com modelos generativos. A ideia central é recuperar informações relevantes de uma base de conhecimento externa ao modelo generativo antes de gerar uma resposta à pergunta, melhorando a precisão e a atualidade das respostas fornecidas por modelos de linguagem. Várias aplicações no domínio da saúde podem se beneficiar da utilização de RAG, como chatbots, assistentes virtuais, sistemas de educação na saúde, suporte à tomada de decisão clínica, entre outros.

Dois macrocomponentes do RAG são cruciais para bons resultados: o *Retriever*, responsável por recuperar fragmentos de texto que contenham informações relevantes para responder à pergunta, e o *Generator*, que se trata de um LLM que recebe essas informações e elabora a resposta, gerando o texto de forma coerente e fortemente fundamentado nos fragmentos recebidos. O *Generator* se baseia na capacidade de aprendizado em contexto (*In-Context Learning* [Brown et al. 2020]) dos LLMs, permitindo que o modelo produza respostas sobre informações que não foram explicitamente incluídas em seu treinamento. Assim, o modelo generativo elabora a resposta com base em um contexto recuperado de uma fonte externa, muitas vezes pertencente a um sistema privado. Esse processo reduz as limitações dos LLMs, garantindo maior precisão e alinhamento com diretrizes clínicas atualizadas.

A implementação de sistemas RAG pode seguir diferentes estratégias como proposto por [Gao et al. 2023]. Os autores destacam *Naïve RAG*, *Advanced RAG* e *Modular RAG* apresentadas na Figura 6.1. Cada estratégia se caracteriza por métodos específicos, com níveis distintos de complexidade, vantagens e limitações [Gao et al. 2023]. Os autores chamam essas estratégias de paradigmas RAG. O *Naïve RAG* é a forma mais básica (indexação, recuperação e geração), sendo simples de implementar, porém com piores resultados e mais suscetível a alucinações e incoerência. O paradigma *Advanced RAG* introduz melhorias em todas as etapas, como otimizações na indexação, reranking, compressão de contexto e busca híbrida, aumentando assim a qualidade das respostas, embora com maior complexidade e custo computacional. Já o *Modular RAG* adota uma arquitetura flexível com módulos especializados permitindo customização para diferentes cenários e domínios, mas exigindo maior esforço técnico. Segundo os autores, cada paradigma apresenta um equilíbrio diferente entre simplicidade, desempenho e adaptabilidade, sendo aplicável conforme o caso de uso.

As estratégias apresentadas na Figura 6.1 trazem uma ideia do nível de complexidade que sistemas RAG podem ter e enfatiza a natureza híbrida dos mesmos, onde resultados são afetados por contribuições de diversos componentes específicos. Assim, apesar dos benefícios significativos do RAG, especialmente em domínios sensíveis como

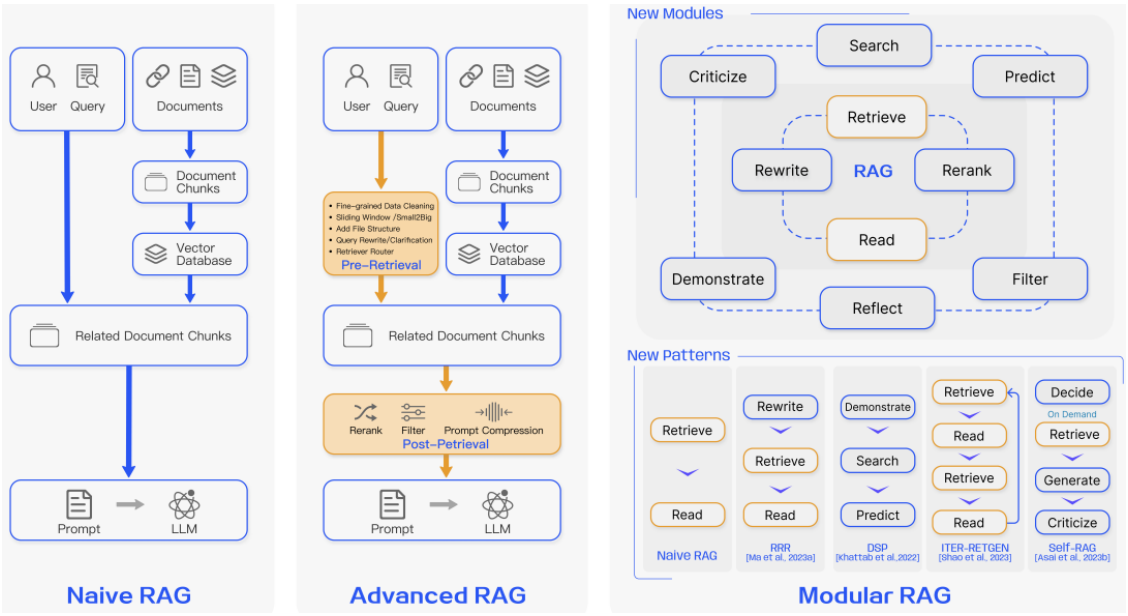

Figura 6.1. Estratégias RAG. Extraído de [Gao et al. 2023]

a saúde, existem diversos desafios na sua implementação e manutenção.

Um dos principais obstáculos está na **qualidade da recuperação**: o sistema pode retornar documentos irrelevantes ou genéricos, deixando de contemplar aspectos essenciais de casos específicos, por exemplo, ao buscar terapias para insuficiência cardíaca em idosos, pode recuperar textos sobre o tratamento em adultos, ignorando particularidades geriátricas. Além disso, o modelo enfrenta dificuldades na **seleção e fusão do contexto**, por exemplo quando precisa combinar informações contraditórias como no caso em que recuperam-se dois documentos sobre COVID-19 que mencionam tratamentos distintos: um sugere corticoterapia, o outro contraindica em determinadas fases. O modelo pode gerar uma resposta que mistura as informações, causando confusão ou erro clínico. Outro desafio técnico é o **limite da janela de contexto** dos modelos de linguagem, que os impede de lidar com documentos extensos, como diretrizes clínicas completas, podendo levar à omissão de dados cruciais.

A **atualização da base de conhecimento** também é um aspecto fundamental, pois o sistema pode se apoiar em evidências desatualizadas, como ocorre em casos que ainda seguem recomendações ultrapassadas sobre o uso de determinados medicamentos. A qualidade final da resposta está diretamente relacionada à qualidade dos documentos disponíveis. Caso o repositório contenha informações defasadas, isso será inevitavelmente refletido nas respostas geradas.

Além disso, há o **custo computacional elevado**, que pode comprometer a usabilidade em contextos de tempo crítico. No RAG, este custo se divide nas duas etapas principais: a recuperação de documentos e a geração de texto. Na primeira, o sistema busca os documentos mais relevantes para uma consulta, o que pode ser feito com métodos como BM25, menos custosos, ou com busca densa baseada em embeddings vetoriais, que exige mais recursos computacionais, especialmente em bases grandes. Na segunda

etapa, os documentos recuperados são usados como contexto para um modelo de linguagem gerar uma resposta, sendo essa fase impactada pelo tamanho do modelo, pela janela de contexto e pela estratégia de decodificação utilizada. Modelos maiores, mais precisos, tendem a ser mais caros em termos de processamento. Adicionalmente, o número de documentos usados como entrada afeta diretamente o tempo e a memória consumidos na geração.

Outro ponto de atenção é a **avaliação da qualidade das respostas**, que não pode se limitar às métricas automáticas; é necessário considerar completude, relevância clínica e impacto potencial da informação gerada. Há o risco da propagação de vieses ou desinformação presentes na base de documentos, o que pode reproduzir práticas ultrapassadas ou discriminatórias, com sérias implicações clínicas e éticas.

Diante desses desafios, torna-se essencial compreender o papel de cada componente do pipeline RAG, a fim de orientar a adoção de estratégias que mitiguem ou eliminem suas limitações. Essa compreensão é particularmente crítica em aplicações de perguntas e respostas na área clínica. Com o objetivo de oferecer uma visão abrangente, este capítulo está estruturado da seguinte forma:

Na Seção 6.2, são apresentados casos de uso de LLMs aplicados à área da saúde, utilizando o conceito de RAG para responder a perguntas clínicas. A Seção 6.3 detalha os diversos componentes do RAG, iniciando pela segmentação de texto (Seção 6.3.1). Na sequência, a Seção 6.3.2 aborda os modelos de embeddings fundamentais para a recuperação densa. A Seção 6.3.3 discute as estratégias de recuperação empregadas pelo *Retriever*, incluindo abordagens tradicionais, densas e híbridas. O *Generator*, responsável pela geração da resposta com base no contexto recuperado, é explorado na Seção 6.3.4, que também apresenta técnicas de geração de texto, engenharia de *prompts* e ajustes de modelos (*fine-tuning*) voltados a domínios sensíveis como o da saúde. A avaliação das respostas e da contribuição de cada componente para o desempenho do sistema é tratada na Seção 6.4. Por fim, a Seção 6.5 conclui o capítulo e destaca oportunidades de pesquisa futuras no contexto da saúde.

6.2. RAG no domínio da saúde

Os LLMs têm o potencial de transformar significativamente a forma como informações médicas são disseminadas. Entretanto, limitações como a dificuldade em acessar dados atualizados e privados e a possibilidade de gerar informações incorretas (alucinações) reduzem sua utilização em contextos clínicos, onde precisão e confiabilidade são cruciais. A abordagem RAG surge como uma alternativa promissora, ao integrar aos LLMs fontes externas de conhecimento. Com isso, os modelos podem acessar informações relevantes e atualizadas fora de seu treinamento original, ampliando sua aplicabilidade em cenários médicos. Embora o uso do RAG na área da saúde ainda esteja em estágio inicial, ele oferece grande potencial para transformar a comunicação e o acesso a informações clínicas [Ng et al. 2025].

Nesta são apresentados trabalhos que aplicam RAG dentro do domínio da saúde. Conceitos relacionados aos componentes RAG que estes trabalhos exploram serão tecnicamente abordados posteriormente na Seção 6.3.

6.2.1. ChatDoctor

[Li et al. 2023] desenvolveram o *ChatDoctor*, um chatbot capaz de responder a perguntas que um paciente faria ao seu médico. A Figura 6.2 representa os principais processos envolvidos na construção do ChatDoctor: (1) coleta de um conjunto de dados contendo conversas entre pacientes e médicos; (2) treinamento do modelo por *fine-tuning*; (3) uso de bases externas para enriquecer as respostas, abordagem que os autores denominam *Autonomous Knowledge Retrieval*; e (4) avaliação da performance do modelo.

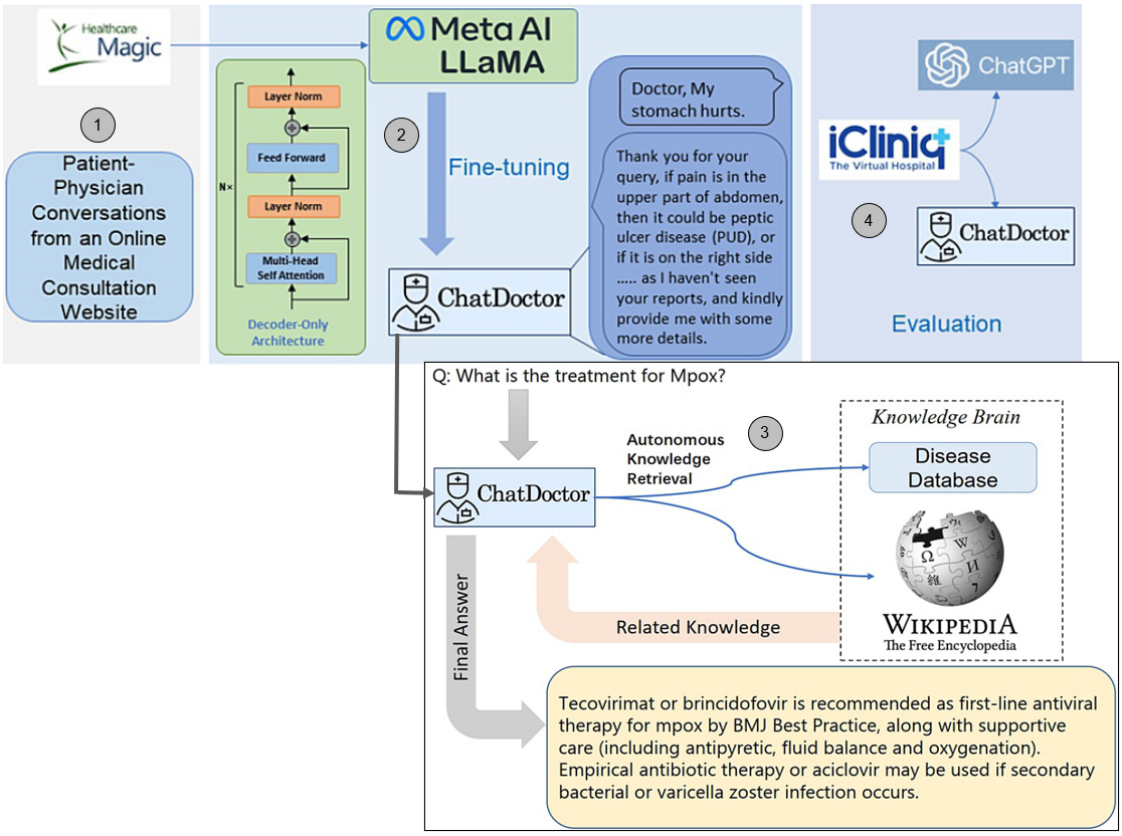

Figura 6.2. Resumo dos processos do Chatdoctor. Adaptado de [Li et al. 2023]

Na etapa (1) referente a coleta de dados, os autores utilizaram um grande conjunto de dados de 100.000 diálogos entre pacientes e médicos, provenientes de uma plataforma de consulta médica online[1] amplamente utilizada. As conversas foram limpas e anonimizadas para respeitar as questões de privacidade. Estes dados foram utilizados na etapa (2) para ajuste do modelo público LLaMA-7B disponibilizado pela Meta.

Além do refinamento do modelo, foi incorporado um mecanismo de recuperação de informações representado na etapa (3) da Figura 6.2. Ao utilizar o *ChatDoctor*, informações de fontes online como a Wikipédia, e dados de bancos de dados médicos offline previamente selecionados, são acessados em tempo real. Essas informações são incorporadas ao que é enviado ao LLM. Para recuperar as informações das bases externas os autores inicialmente usaram o LLM para identificar palavras chaves da pergunta

[1] www.healthcaremagic.com

como pode ser observado na Figura 6.3 no *prompt A*. Os termos identificados são usados então na busca por documentos que contenham os mesmos. Os textos foram divididos em seções iguais num tamanho que o LLM conseguiria administrar. As seções contendo as palavras-chaves são ordenadas pelo número de ocorrências destes termos, e as N primeiras seções são adicionadas ao *prompt*. O modelo *ChatDoctor* usa as cinco primeiras ($N = 5$), usando o *prompt B* da Figura 6.2 para selecionar as informações pertinentes. Por fim, a pergunta e as informações selecionadas são adicionadas ao *prompt C*, que solicita a geração da resposta final.

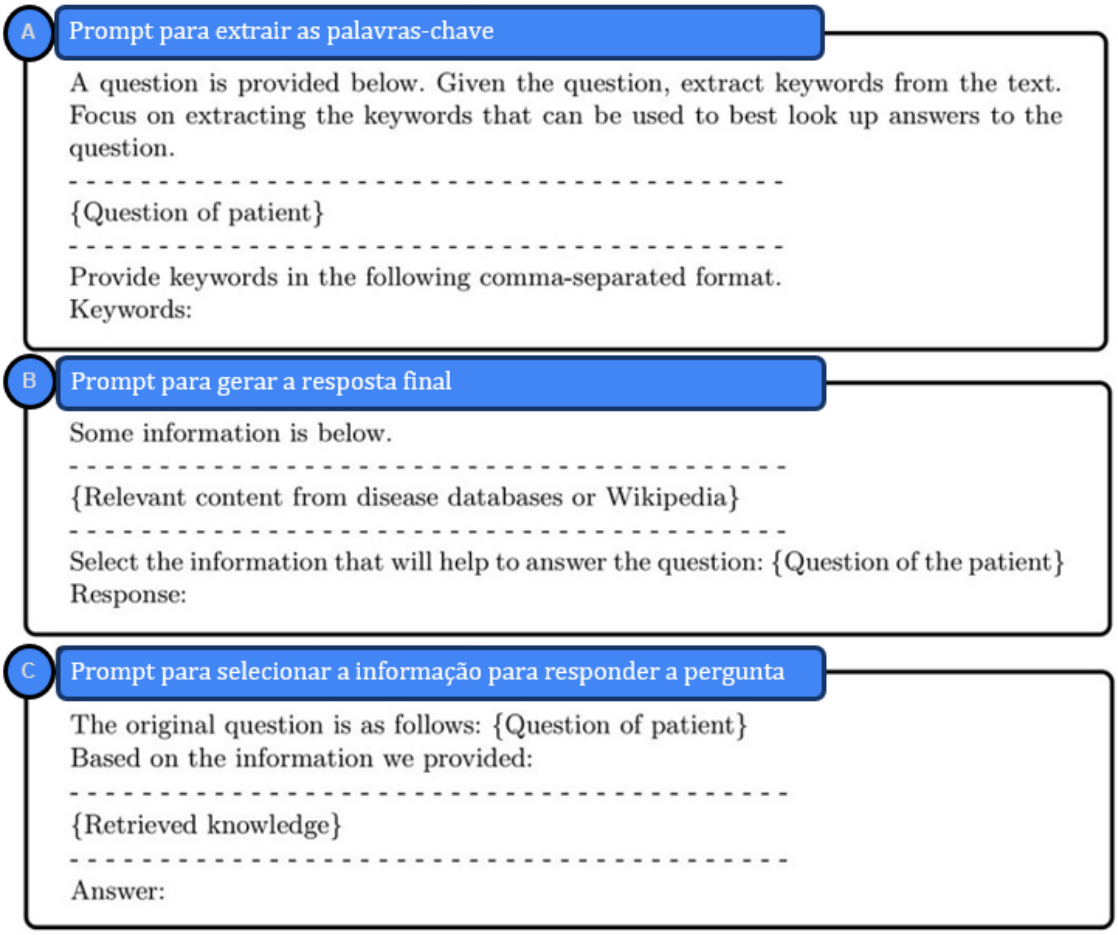

Figura 6.3. *Prompts* usados no *Chatdoctor*. Adaptado de [Li et al. 2023]

Para uma avaliação quantitativa do desempenho do *ChatDoctor*, representado na etapa (4) da Figura 6.2, foram utilizadas aproximadamente 10 mil conversas adicionais de um site independente de consulta médica online, o iCliniq[2]. Tratam-se de perguntas efetuadas por pacientes e as respectivas respostas de médicos que servem como referência para avaliação. Foram geradas as respostas usando o *ChatDoctor* e o ChatGPT (usando o modelo gpt-3.5-turbo). Para avaliar a similaridade semântica entre as respostas dos modelos e as respostas de referência, a métrica BERTScore [Zhang et al. 2020] foi utilizada. O *ChatDoctor* apresentou desempenho significativamente superior ao ChatGPT.

[2]https://www.icliniq.com/

Além disso, os autores destacam sua capacidade de lidar com doenças emergentes. Para demonstrar esse aspecto, o modelo foi testado com uma variedade de consultas médicas sobre temas recentes. Um exemplo é uma pergunta relacionada à "Varíola dos Macacos", termo recém designado pela Organização Mundial da Saúde (OMS) em 28 de novembro de 2022. Por se tratar de uma designação recente, a versão do ChatGPT utilizada à época não foi capaz de reconhecer o termo, ao contrário do *ChatDoctor*.

6.2.2. Almanac

Outro framework para RAG aplicado a dados clínicos é o *Almanac* [Zakka et al. 2024], que combina a capacidade dos LLMs com recuperação aumentada, sendo projetado para apoiar a tomada de decisões clínicas com segurança e precisão. O *Almanac* é composto por diversos componentes, conforme ilustrado na Figura 6.4. A pergunta feita no início do fluxo (1) é simplificada com o auxílio de um LLM (GPT-4), que gera termos ou palavras-chave para pesquisa. Esses termos são inseridos em um navegador especializado (2), que acessa exclusivamente fontes verificadas, como PubMed, UpToDate e BMJ Best Practice. Apenas fontes previamente registradas no *Almanac* como confiáveis são consultadas durante o processo.

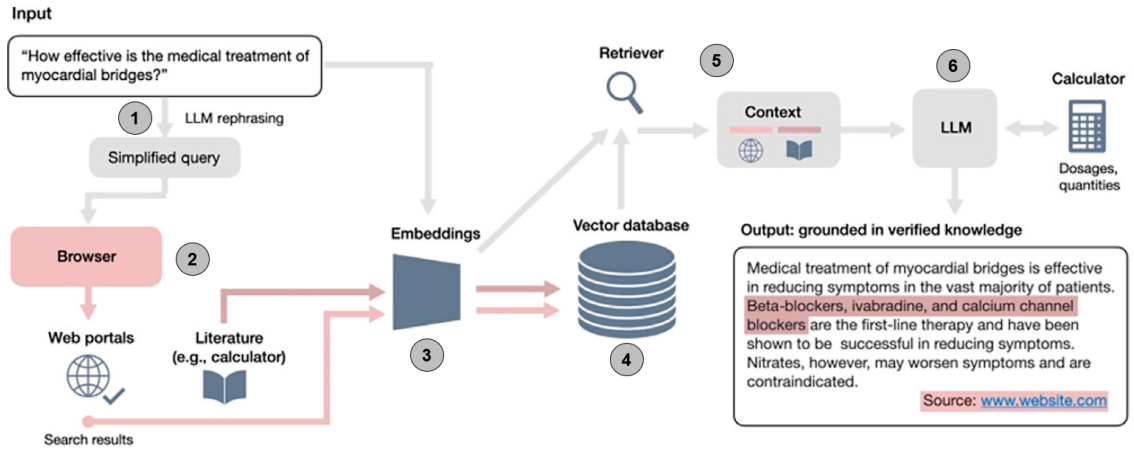

Figura 6.4. Visão geral do *Almanac*. Adaptado de [Zakka et al. 2024]

Os documentos recuperados pelo *Almanac* são divididos em fragmentos de 1000 tokens. As representações numéricas (embeddings) dos fragmentos são geradas, correspondente à etapa (3) na Figura 6.4. Cada fragmento tem os embeddings gerados separadamente usando o modelo text-embedding-ada-002 da OpenAI e armazenados no banco de dados vetorial Qdrant[3] (4). O Qdrant é então utilizado como *Retriever* para recuperar conteúdo relevante à pergunta na base de dados usando a similaridade de cosseno (5). O Qdrant também é inicializado com mais de 500 calculadoras clínicas. Essas calculadoras são obtidas diretamente do MDCalc[4] e suas indicações e instruções clínicas são usadas como metadados para recuperação. O LLM é usado em vários momentos com diferentes *prompts*. Alguns exemplos estão apresentados na Tabela 6.1. Inicialmente o LLM é usado para identificar os termos-chave na consulta (1) e depois para sintetizar as respostas finais

[3]https://qdrant.tech/
[4]https://www.mdcalc.com/

com citações no texto e para utilizar as calculadoras relevantes dependendo dos termos da consulta (6). Os autores não esclarecem quanto à persistência dos documentos indexados no Qdrant, ou seja, por quanto tempo os fragmentos indexados são mantidos na base.

Tabela 6.1. Exemplos de *prompts* usados nas diferentes etapas do *Almanac*.

Etapa	Prompt
Simplificação da consulta	`Given question (Q) convert it to a simple Google search term.`
Geração da Resposta completa	`Generate a thorough and concise answer for a given question (Q) on the basis of the provided context (C). If you are asked to calculate a value, output the final equation in Python code. Use an unbiased and journalistic tone. If you cannot find a relevant answer, write "I apologize but there doesn't seem to be a clear answer to your question based on my sources ... Answer the question based on your own knowledge." Cite sources as [1] or [2] or [3] after each sentence to back up your answer (Ex: Correct: [1], Correct: [2][3], Incorrect: [1, 2]).`
Sumarização da Resposta	`Given the question (Q), context (C), and output (O), generate a thorough and concise answer for the given question (Q).`

Além do framework, os autores compilaram um novo benchmark, o ClinicalQA, com 314 perguntas clínicas abertas desenvolvidas por 10 especialistas. Diversas especialidades médicas estão contempladas no dataset, com tópicos que vão desde diretrizes de tratamento até cálculos clínicos. Para cada pergunta, foram geradas as respostas usando *Almanac*, Bing, ChatGPT e Bard. Cada especialista avaliou as respostas geradas pelos quatro modelos dentro de três dimensões apresentadas na Tabela 6.2. Para cada pergunta dentro de cada dimensão o especialista numerou os modelos de 1 (melhor) a 4 (pior). Ao final, foi efetuada a média de cada modelo e *Almanac* superou significativamente os demais em todas as dimensões. Também foi avaliada a veracidade das citações geradas nas respostas, pois todos os modelos foram solicitados a fornecer as referências, neste aspecto *Almanac* também superou os demais.

6.2.3. Triagem para ensaios clínicos

A triagem de pacientes para ensaios clínicos é tradicionalmente realizada de forma manual, sendo propensa a erros e altamente demorada. Buscando minimizar estes problemas, [Unlu et al. 2024] propõem RECTIFIER (*RAG-Enabled Clinical Trial Infrastructure for Inclusion Exclusion Review*), um sistema RAG baseado no GPT-4 visando melhorar a precisão, a eficiência e a relação custo-benefício da determinação da elegibilidade para ensaios clínicos usando dados não estruturados de prontuários eletrônicos. O caso de

Tabela 6.2. Métricas usadas na avaliação do *Almanac* e demais modelos

Factualidade	A resposta está de acordo com as práticas padrão e o consenso estabelecido pelos órgãos de autoridade em sua área de atuação? Se aplicável, a resposta contém etapas de raciocínio corretas?
Completude	A resposta aborda todos os aspectos da questão? A resposta omite algum conteúdo importante? A resposta contém algum conteúdo irrelevante?
Preferência	Qual resposta você preferiu no geral?

uso trata da triagem de pacientes para o estudo COPILOT-HF (focado em pacientes com insuficiência cardíaca), onde muitos critérios de elegibilidade dependem de prontuários clínicos em texto livre.

Para avaliar a elegibilidade dos pacientes, os pesquisadores desenvolveram um conjunto de perguntas a serem respondidas automaticamente com base em anotações clínicas existentes dos últimos dois anos. As anotações foram divididas em fragmentos com menos de 1000 tokens usando a estratégia de segmentação recursiva do LangChain [5] para preservar o contexto e evitar o truncamento de frases ou de palavras no meio. Em seguida foram geradas as representações vetoriais numéricas (embeddings) para cada fragmentos das anotações clínicas usando o modelo text-embedding-ada-002 da OpenAI. Para otimizar a recuperação durante a etapa de perguntas e respostas, utilizou-se a biblioteca AI Similarity Search (FAISS) do Facebook[6].

Foram criados *prompts* relacionados aos critérios alvo (perguntas de elegibilidade) para cada paciente. As perguntas foram transformadas em embeddings, que serviram como consultas de pesquisa no banco de dados de vetores, usando LangChain. As buscas recuperaram os três fragmentos das anotações clínicas mais relevantes com base na similaridade semântica de cada pergunta, juntamente com os links para as anotações da fonte original. O modelo não considerou a atualidade das anotações e simplesmente recuperou as mais relevantes. Esses fragmentos recuperados foram então combinados com uma instrução, conforme o exemplo da Tabela 6.3, que trata dos critérios exclusivos. A instrução e os fragmentos das anotações clínicas foram enviados para o modelo GPT-4 com temperatura de 0, o que gerou respostas binárias de "Sim"ou "Não".

Os pacientes foram divididos em 100 (desenvolvimento) e 282 (validação) para investigação dos melhores *prompts* a utilizar, e 1894 (conjunto de teste), dos quais 1509 possuíam anotações clínicas suficientes e que foram de fato utilizados na avaliação. Para comparar o desempenho do RECTIFIER com o desempenho da equipe do estudo para triagem, um clínico especialista efetuou uma revisão cega dos pacientes do conjunto de testes e respondeu às perguntas dos critérios-alvo (Sim/Não) sem acessar as respostas do RECTIFIER nem da equipe de estudo, estabelecendo assim respostas de referência (*gold standard*).

As respostas da equipe do estudo e do RECTIFIER foram bastante alinhadas com o *gold standard* com precisão variando entre 97,9% e 100% para o RECTIFIER e entre

[5]http://langchain.com/

[6]https://github.com/facebookresearch/faiss

Tabela 6.3. Exemplo do *prompt* usado para as perguntas dos critérios de exclusão dos pacientes

```
Can you please answer each of the following 12 questions
based on the information in the clinical notes with
only "Yes"or "No"?  Please return the answers in a comma
separated list.
1) Does the patient have unrepaired severe aortic stenosis,
or have unrepaired severe aortic valve insufficiency?
Repair includes aortic valve surgery and TAVR.
2) Does the patient have a history of known amyloid heart
disease?
3) Does the patient currently have WHO Group 1 pulmonary
arterial hypertension on disease-specific therapies like
Ambrisentan, Bosentan, Epoprostenol, Treprostinil, Iloprost
(do not include if the patient is on ONLY Sildenafil or
Tadalafil as disease-specific therapies)?
4) Is the patient currently getting chemotherapy or
hormonal therapy due to an active malignancy?
5) Is the patient currently receiving or will receive
hospice care?  Answer only for the patient, not for the
family members.
6) Does the patient have a history (Hx) of solid organ
transplant, being evaluated for transplant, or currently
on wait list above at the UNOS status level above 4?
7) Does the patient currently use a Ventricular Assist
Device?
8) Does the patient have a history of established
hypertrophic cardiomyopathy?
9) Does the patient have a history of Type 1 Diabetes?
10) Is the patient currently undergoing dialysis?
11) Is the patient currently pregnant or breastfeeding?
12) Does the patient have a history of Congenital Heart
Disease?
```

91,7% e 100% para a equipe do estudo. O RECTIFIER apresentou desempenho superior ao da equipe do estudo na determinação de insuficiência cardíaca sintomática, com precisão de 97,9% versus 91,7%. No geral, a sensibilidade e a especificidade para determinar a elegibilidade do paciente com o RECTIFIER foram de 92,3% e 93,9%, respectivamente, e 90,1% e 83,6% com a equipe do estudo.

6.2.4. Procedimentos Operacionais Padrão

Os profissionais de saúde frequentemente devem seguir procedimentos operacionais padrão (POPs) que estabelecem, por exemplo, protocolos internos na comunicação e disseminação de informações [Kuriki P. and R. 2024]. Os métodos de busca tradicionais têm dificuldades com consultas formuladas em linguagem natural, especialmente em meio a

grandes volumes de dados, o que destaca a necessidade de um sistema de recuperação mais eficaz. Neste contexto, a abordagem RAG aprimora o gerenciamento de informações, permitindo uma recuperação precisa e baseada em linguagem natural.

[Kuriki P. and R. 2024] apresentaram, durante a SIIM24[7] (Reunião Anual da Sociedade de Informática de Imagem em Medicina), um chatbot clínico baseado na abordagem RAG. O sistema demonstrou melhorias na precisão da recuperação de Procedimentos Operacionais Padrão (POPs), ilustrando seu potencial de aplicação na indústria farmacêutica para garantir conformidade, reduzir erros e agilizar atualizações de protocolos. Para sua implementação, foi utilizado o modelo de linguagem Mistral-7B, e o fluxo geral do sistema pode ser visualizado na Figura 6.5.

Os textos das POPs foram convertidos em embeddings que foram armazenados em um banco de dados vetorial chamado SOP2vec, permitindo a busca por similaridade. Uma interface web de chatbot permite que os usuários interajam com os documentos usando linguagem natural. As perguntas são convertidas em embeddings que são comparadas com o banco de dados vetorial, recuperando fragmentos de contexto que são adicionados ao *prompt*. O *prompt* é enviado a um modelo Mistral-7B implantado localmente, gerando uma resposta para o usuário. Os autores escolheram a abordagem local por questões de desempenho e segurança no manuseio de informações sensíveis.

Os autores relatam a implementação de um processo de avaliação automático usando GPT-4. Foram geradas de 10 a 20 perguntas por documento pertencentes a um conjunto de POPs. Essas perguntas foram aplicadas ao *pipeline* RAG+LLM, e cada resposta foi avaliada pelo GPT-4 quanto à acurácia da recuperação e à relevância da resposta. Segundo os autores, esse processo permitiu a identificação de falhas, levando a melhorias significativas, por exemplo, a otimização dos embeddings, refatoração de perguntas, aplicação de reranking, melhorias em metadados e correções de erros nas POPs.

6.3. Componentes do RAG

Nesta seção, serão apresentados os componentes e processos da abordagem RAG. O RAG tem duas macroetapas principais: a recuperação (*Retriever*) e a geração (*Generator*, também conhecida como *Reader* na literatura, mas referida neste capítulo como *Generator*) , conforme ilustrado na Figura 6.6.

O *Retriever* engloba todas as tarefas relacionadas à obtenção de contexto adicional de bases externas que possam auxiliar a responder à pergunta. O *Generator* recebe a pergunta e um conjunto de documentos ou partes de documentos (o contexto adicional) e gera a resposta.

O conjunto de documentos funciona como uma espécie de extensão (aumento) do conhecimento paramétrico do LLM, pois explora o aprendizado baseado no contexto (*In-Context Learning*, ICL). ICL refere-se à capacidade de modelos de linguagem, como o GPT, de aprender e realizar tarefas apenas a partir do contexto fornecido na entrada (*prompt*), sem a necessidade de ajustar os pesos do modelo por meio de um novo treinamento. Assim, em vez de treinar o modelo novamente, são fornecidos exemplos no próprio *prompt* (*few-shot learning*), e o modelo utiliza esses exemplos para gerar a res-

[7]https://siim.org/

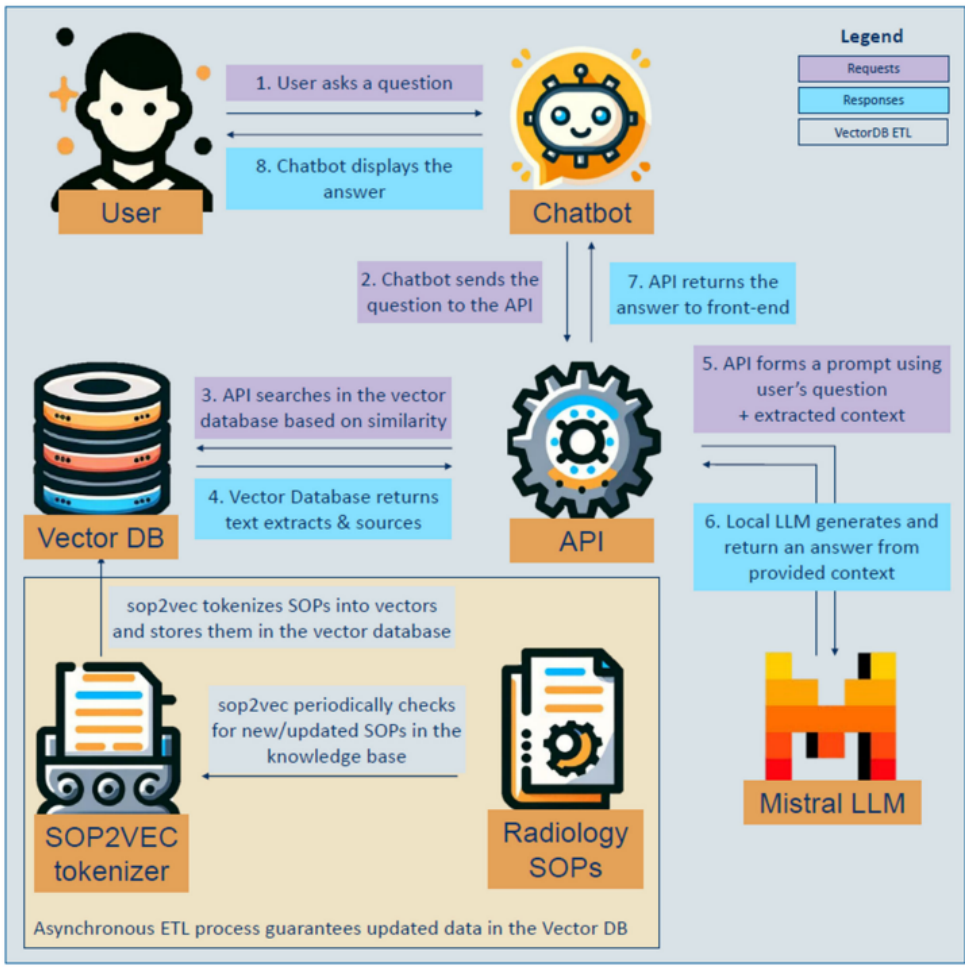

Figura 6.5. RAG para Procedimentos Operacionais Padrão (POPs) na saúde. Extraído de [Kuriki P. and R. 2024]

posta adequada [Brown et al. 2020].

Existem vários componentes dentro das macroetapas do *Retriever* e do *Generator*, utilizadas quando o sistema está "em produção". Mas também há uma série de etapas no estágio pré-produção, envolvendo especialmente o preparo dos documentos usados pelo RAG, como pode ser observado na Figura 6.7 que apresenta em mais detalhes um *pipeline* RAG. Cada etapa será explicada de forma geral nesta seção e mais detalhadamente nas seções subsequentes.

6.3.1. Segmentação do texto

Dividir grandes volumes de texto em blocos menores e de mais fácil processamento é uma prática conhecida como segmentação. Esse processo é amplamente conhecido pelo termo em inglês *chunking* e assim nos referiremos a ele neste capítulo. Embora seja uma etapa fundamental em diversas aplicações de processamento de linguagem natural e recuperação de informação, muitas vezes é deixada em segundo plano. Na Figura 6.7 o *chunking* faz parte da etapa de pré-produção, onde os documentos da base de dados serão processados e segmentados para posterior indexação. Um *chunk* se refere a um segmento

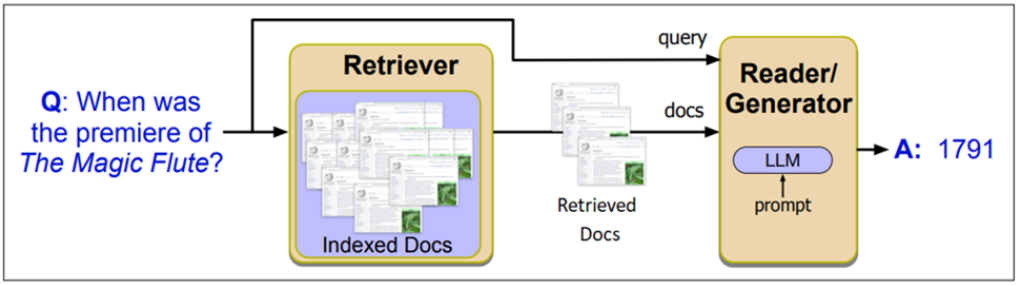

Figura 6.6. As macroetapas do RAG. Extraído de [Jurafsky and Martin 2025]

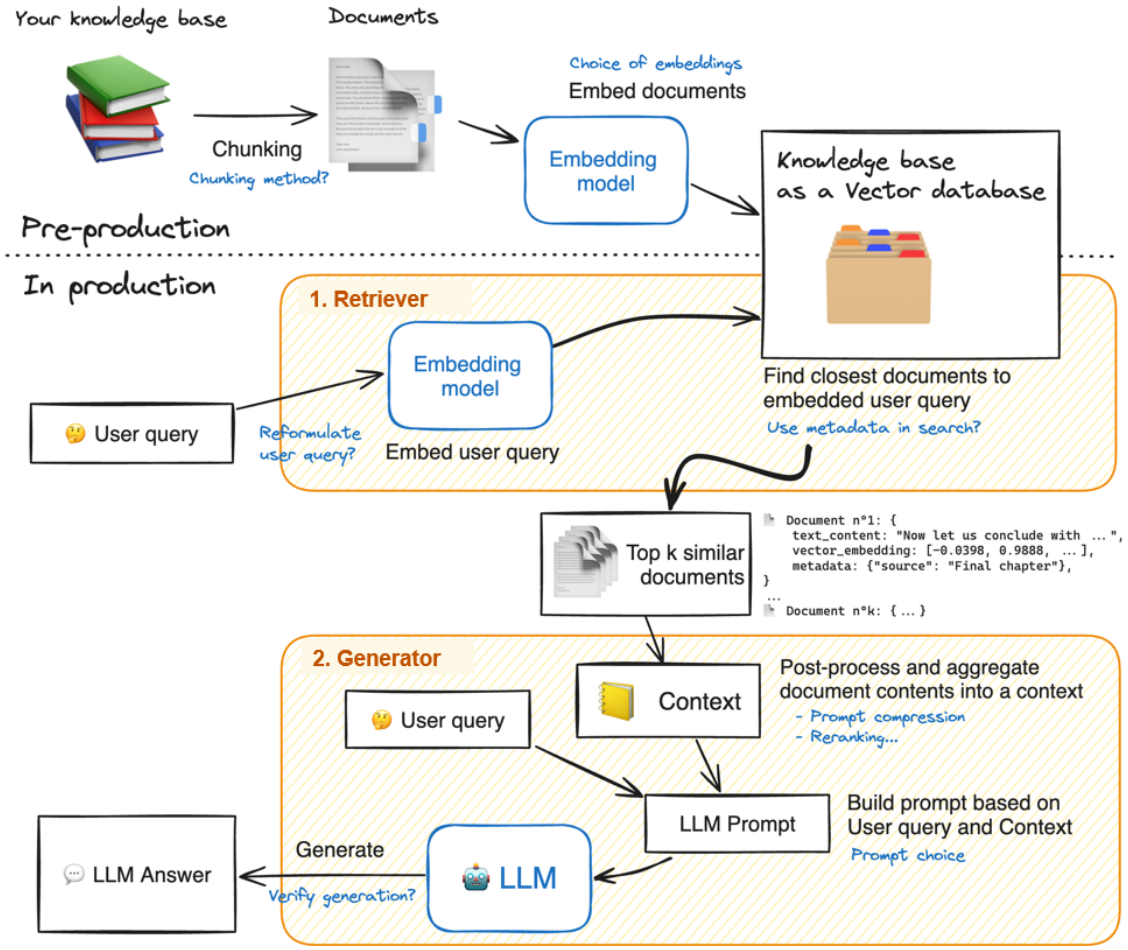

Figura 6.7. As macroetapas do RAG. Adaptado de [Huggingface 2024]

de texto.

A granularidade na qual um documento é segmentado desempenha um papel essencial, pois estratégias de *chunking* ineficazes podem levar a fragmentos com contexto incompleto ou excesso de informações irrelevantes, o que prejudica o desempenho dos modelos de recuperação [Duarte et al. 2024b]. Ao final, *chunks* adequados, coesos e semanticamente significativos melhoram a precisão da busca aumentando a probabilidade de respostas mais coerentes.

Neste contexto, uma das escolhas a fazer é o tamanho máximo do *chunk*, que impacta limites das janelas de contexto dos modelos usados tanto para representação do texto (modelos para geração de embeddings vistos na Seção 6.3.2) como dos LLMs na geração da resposta final (Seção 6.3.4.1). A janela de contexto é o tamanho máximo de entrada dos modelos, que é quantificada em número de tokens. Tokens podem ser palavras ou sub-palavras e representam as unidades mínimas de texto em que uma sequência textual é dividida para ser processada pelo modelo. Cada modelo utiliza um tokenizador (algoritmo responsável por dividir o texto em tokens) específico e portanto o número de tokens de um mesmo segmento de texto pode ser diferente para cada modelo. O E5[8] é um exemplo de modelo para extração de embeddings usado na recuperação densa (ver Seção 6.3.2) que possui 512 tokens de entrada, enquanto o modelo text-embedding-3-large[9] da Open AI tem 8.191 tokens.

As janelas de contexto dos LLMs usados na resposta também precisam ser consideradas. O GPT-4o possui uma janela de 128 mil tokens, o Mistral-7B-Instruct-v0.2 tem 32 mil tokens, já o LLamma 2 e GPT-3.5-turbo ambos tem 4.096 tokens. A ideia do RAG é recuperar *chunks* relevantes para a pergunta visando compor o contexto que será enviado ao LLM juntamente com a pergunta. Dessa forma, *chunks* muito longos podem trazer problemas na entrada dos modelos.

Uma abordagem bastante comum e simples é utilizar um comprimento fixo de *chunk* e o texto de um documento é dividido em blocos com no máximo esse tamanho. Geralmente essa técnica é usada considerando uma sobreposição. Na Figura 6.8 podem ser observadas diferentes estratégias de *chunking* para o texto "*A gripe é uma infecção aguda do sistema respiratório, provocado pelo vírus da influenza, com grande potencial de transmissão. Existem quatro tipos de vírus influenza/gripe: A, B, C e D. O vírus influenza A e B são responsáveis por epidemias sazonais, sendo o vírus influenza A responsável pelas pandemias.*". A estratégia "A"usa um tamanho de 15 tokens com nenhum recobrimento. A estratégia "B"usa o mesmo número de tokens e uma sobreposição de 20% (os 3 primeiros tokens do *chunk* são os 3 últimos do *chunk* imediatamente anterior). Já na estratégia "C"os *chunks* são separados por frases.

Uma estratégia de *chunking* comumente empregada em sistemas RAG é a fragmentação recursiva, cujo objetivo é preservar a integridade semântica dos trechos recuperáveis. O método utiliza marcadores sintáticos (por exemplo vírgulas, pontos finais e quebras de parágrafo) como delimitadores naturais para particionar o conteúdo. A recursividade está no fato de que se o trecho ainda for grande, tenta-se quebrar novamente em unidades menores, respeitando limites linguísticos naturais. O processo continua até que os *chunks* estejam dentro do limite de tamanho definido. Isso ajuda a manter coerência semântica dentro de cada segmento e evita que partes relacionadas fiquem separadas em *chunks* diferentes. Frameworks como LangChain disponibilizam *chunking* recursivo[10]. Por ser computacionalmente eficiente, mostra-se apropriado para *pipelines* em larga escala, sobretudo quando aplicado a textos bem estruturados. Ao preservar a coesão semântica entre segmentos adjacentes, essa abordagem contribui para a geração

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

[9]https://platform.openai.com/docs/guides/embeddings/embedding-models#embedding-models

[10]https://python.langchain.com/v0.1/docs/modules/data_connection/document_transformers/recursive_text_splitter/

A) 15 tokens 0% de recobrimento:

1	A gripe é uma infecção aguda do sistema respiratório, provocado pelo vírus da influenza
2	, com grande potencial de transmissão. Existem quatro tipos de vírus influenza/gripe
3	: A, B, C e D. O vírus influenza A e B
4	são responsáveis por epidemias sazonais, sendo o vírus influenza A responsável pelas pandemias.

B) 15 tokens e 20% de recobrimento:

1	A gripe é uma infecção aguda do sistema respiratório, provocado pelo vírus da influenza
2	vírus da influenza, com grande potencial de transmissão. Existem quatro tipos de vírus
3	tipos de vírus influenza/gripe: A, B, C e D.
4	e D. O vírus influenza A e B são responsáveis por epidemias sazonais,
5	epidemias sazonais, sendo o vírus influenza A responsável pelas pandemias.

C) Separando por frases:

1	A gripe é uma infecção aguda do sistema respiratório, provocado pelo vírus da influenza, com grande potencial de transmissão.
2	Existem quatro tipos de vírus influenza/gripe: A, B, C e D.
3	O vírus influenza A e B são responsáveis por epidemias sazonais, sendo o vírus influenza A responsável pelas pandemias.

Figura 6.8. Exemplos de estratégias de *chunking*

de representações vetoriais mais informativas e, consequentemente, para a melhoria da qualidade da recuperação e da geração de respostas [Lee et al. 2021].

No trabalho de [Kamradt 2024], é adotada uma abordagem baseada em embeddings densos para identificar *chunks* semanticamente próximos. O processo inicia com a segmentação inicial do texto por meio de uma técnica de fragmentação recursiva, que delimita sentenças de forma estruturada. Em seguida, para cada *chunk*, são geradas representações vetoriais utilizando uma janela deslizante composta por múltiplas sentenças consecutivas. A partir dessa sequência vetorial organizada, o método percorre iterativamente o conjunto, comparando janelas adjacentes com o objetivo de detectar transições semânticas e, assim, estabelecer pontos adequados de segmentação.

A abordagem denominada *LumberChunker* [Duarte et al. 2024a] explora a capacidade de raciocínio dos modelos de linguagem para realizar a segmentação de textos de forma a maximizar a independência semântica entre os *chunks* gerados. Nesse método, o texto é inicialmente separado em parágrafos, os quais são submetidos iterativamente a um LLM. O modelo recebe a tarefa de identificar o ponto em que ocorre uma mudança substancial de conteúdo em relação ao que foi previamente apresentado. Os parágrafos anteriores a essa transição são agrupados e definidos como um *chunk*. O processo é repetido em sequência até que todo o conteúdo seja organizado.

A estratégia de mistura de *chunkers* (*Mixture-of-Chunkers*, MoC) [Zhao et al. 2025] busca otimizar o *chunking* em sistemas RAG usando uma rede de especialistas leves (meta-*chunkers*) gerenciados por um roteador sensível à granularidade. Ele se destaca por gerar regras de *chunking* (expressões regulares) em vez de texto completo, o que reduz o custo computacional, e utiliza um algoritmo de pós-processamento para garantir a precisão da extração dos blocos. Os experimentos mostraram que o MoC melhora o desempenho do RAG em comparação com métodos anteriores.

[Ke et al. 2024] implementaram um sistema RAG com GPT4-o para diretrizes pré-operatórias. As respostas do RAG foram comparadas a respostas geradas por humanos atingindo bons resultados. Esse trabalho utilizou embeddings do modelo text-embedding-ada-002 (modelo com entrada de 8191 tokens). Foi utilizado o *chunking* recursivo do Langchain configurando tamanho máximo de 1000 tokens e um overlap de 100 tokens.

6.3.2. Modelos de embeddings

De forma geral embeddings são vetores de números reais que visam representar dados discretos (palavras, itens, usuários ou imagens). Em aprendizado profundo, os embeddings podem ser definidos como representações vetoriais densas (a maioria ou praticamente todos os valores dentro do vetor são diferentes de zero) e de dimensão fixa utilizadas para codificar os dados em um espaço contínuo de alta dimensionalidade. O objetivo é capturar relações semânticas entre os dados que estes vetores representam, de forma que itens com significados ou comportamentos semelhantes fiquem próximos entre si nesse espaço vetorial.

Métodos como Word2Vec [Mikolov et al. 2013] e GloVe [Pennington et al. 2014] propuseram técnicas para o aprendizado de embeddings, atribuindo um vetor fixo a cada palavra no vocabulário do corpus utilizado durante o treinamento. Esses vetores são aprendidos automaticamente durante o treinamento destes modelos e capturam muito bem a sinonímia, entretanto não resolvem a polissemia (uma mesma palavra pode ter diferentes sentidos, como por exemplo a palavra "banco": banco da praça, banco de areia, banco como instituição financeira, banco de dados, banco do verbo bancar).

Nos modelos baseados na arquitetura Transformer [Vaswani et al. 2017], como o BERT [Devlin et al. 2019], os embeddings passaram a ser chamados de contextuais, pois o vetor atribuído a uma palavra pode variar conforme o contexto em que ela aparece, resolvendo assim o problema da polissemia. A capacidade destes modelos baseados em atenção de gerar embeddings contextuais reside em seu mecanismo de atenção que simula a forma como os humanos focam em diferentes partes de uma informação ao processá-la. Dessa forma, o modelo pondera a importância de cada palavra em uma sequência (por exemplo, uma frase) ao gerar a representação vetorial de outra palavra na mesma sequência. Em vez de tratar todas as palavras igualmente, a atenção permite que o modelo foque nas palavras mais relevantes do contexto da palavra que está sendo codificada.

Nos sistemas RAG, os modelos de embeddings são utilizados em dois momentos conforme apresentado na Figura 6.7: nos processos de pré-produção, onde são usados para gerar os vetores de toda a base de dados que será consultada posteriormente; e em produção, na codificação dos vetores da pergunta que serão utilizadas pelo *Retriever* para recuperar vetores semanticamente similares na base de dados.

Muitos modelos de embeddings modernos (como SBERT[11], E5[12], BGE[13], GTE[14]) são construídos sobre a mesma arquitetura Transformer que os LLMs. Eles são "gran-

[11]https://huggingface.co/sentence-transformers
[12]https://huggingface.co/intfloat/multilingual-e5-large
[13]https://huggingface.co/BAAI/bge-large-en-v1.5
[14]https://huggingface.co/Alibaba-NLP/gte-modernbert-base

des"no sentido de terem muitos parâmetros e treinados em vastos volumes de dados textuais. Os modelos de embeddings "modelam"a linguagem para criar suas representações. Assim, teoricamente, qualquer modelo que aprenda a partir de dados de linguagem e capture aspectos dela poder ser considerado um "modelo de linguagem". Neste capítulo, quando usamos o termo LLMs estamos nos referindo aos modelos utilizados na geração da resposta em sistemas RAG.

Existem diversos modelos de embeddings disponíveis. A capacidade semântica destes modelos são avaliadas por meio de tarefas como recuperação de informação, agrupamento, classificação, etc. O MTEB (*Massive Text Embedding Benchmark*) é um benchmark reconhecido onde é possível encontrar o ranking de modelos de embeddings. A maior parte dos modelos disponibilizados até este momento priorizam a língua inglesa, existindo alguns multilíngues. No painel do MTEB é possível verificar o desempenho destes modelos em datasets do domínio médico[15]. São encontrados datasets do domínio usados para avaliar estes modelos, como pode ser verificado na Tabela 6.4.

Tabela 6.4. Datasets Médicos disponíveis no MTEB

Dataset	**Tipo de Tarefa**	**Linguagens**	**Domínios**
CMedQAv2-reranking	Reranking	mandarim	Médico
CUREv1	Retrieval	inglês, árabe, chines, francês, coreano, russo, espanhol, vietnamita	Médico, Acadêmico
CmedqaRetrieval	Retrieval	mandarim	Médico
MedicalQARetrieval	Retrieval	inglês	Médico
Medrxiv-ClusteringP2P.v2	Clustering	inglês	Acadêmico, Médico
Medrxiv-ClusteringS2S.v2	Clustering	inglês	Acadêmico, Médico
NFCorpus	Retrieval	inglês	Médico, Acadêmico
PublicHealthQA	Retrieval	inglês, árabe, chines, francês, coreano, russo, espanhol, vietnamita	Médico, Governamental, Web
SciFact	Retrieval	inglês	Acadêmico, Médico
TRECCOVID	Retrieval	inglês	Médico, Acadêmico

O dataset PublicHealthQA, por exemplo, é utilizado para medir a capacidade dos modelos em recuperar informações relevantes em contextos médicos e de saúde pública. O dataset contem perguntas em oito línguas (inglês, árabe, chines, francês, coreano, russo, espanhol, vietnamita). Na Tabela 6.5 alguns exemplos de perguntas e respostas em inglês e espanhol.

No RAG, os modelos de embeddings convertem os fragmentos de texto (chunks)

[15]https://mteb-leaderboard.hf.space/?benchmark_name=MTEB%28Medical%2C+v1%29

Tabela 6.5. Exemplos do dataset PublicHealthQA

Pergunta	Resposta
What temperature kills the virus that causes COVID-19?	Generally coronaviruses survive for shorter periods at higher temperatures and higher humidity than in cooler or dryer environments. However, we don't have direct data for this virus, nor do we have direct data for a temperature-based cutoff for inactivation at this point. The necessary temperature would also be based on the materials of the surface, the environment, etc. Regardless of temperature please follow CDC's guidance for cleaning and disinfection.
¿Cómo debo preparar a mis hijos por si se produce un brote de COVID-19 en nuestra comunidad?	Los brotes pueden ser estresantes tanto para los adultos como para los niños. Hable con sus hijos acerca del brote, intente mantener la calma y recuérdeles que se encuentran a salvo. Si lo considera apropiado, explíqueles que la mayoría de los casos COVID-19 parecen ser leves. Los niños responden de manera diferente a las situaciones estresantes que los adultos. Los CDC ofrecen recursos para conversar con los niños acerca del COVID-19.

em vetores dentro de um espaço vetorial. Estes vetores são usados nas buscas semânticas baseadas numa métrica de similaridade, o que é discutido na Seção 6.3.3. Para isso ocorrer, os vetores precisam ser pré-processados e armazenados para estarem aptos a serem utilizados nas buscas. Existem uma série de desafios técnicos na gestão e uso dos vetores, entre eles estão [Pan et al. 2024]:

- Ambiguidade da similaridade semântica: é difícil definir de forma inequívoca o que é "semelhante", e diferentes funções de similaridade podem produzir rankings divergentes.
- Custo computacional elevado: comparações de similaridade são proporcionalmente caras.
- Ausência de propriedades estruturais: diferente de dados relacionais, vetores não são naturalmente ordenáveis.
- Consultas híbridas: dificuldade de combinar vetores e atributos tradicionais de forma eficiente.

Existem diversas opções de banco de dados vetoriais como Pinecone[16], Qdrant[17], CrhomaDB[18], Vespa[19], Milus[20], e ferramentas de busca que suportam o armazento veto-

[16]https://www.pinecone.io/
[17]https://qdrant.tech/
[18]https://www.trychroma.com/
[19]https://vespa.ai/
[20]https://milvus.io

rial como ElasticSearch[21], OpenSearch[22], etc. Segundo [Pan et al. 2024], para aplicações baseadas em RAG, a escolha de um sistema de banco de dados vetorial (*Vector Database Management Systems*, VDBMS) deve considerar uma combinação de requisitos que garantam desempenho, flexibilidade e integração com modelos de linguagem. Um VDBMS ideal para RAG precisa apresentar alta eficiência na execução de buscas por vizinhos mais próximos, tanto exatas (*k-Nearest Neighbors*, k-NN) quanto aproximadas (*Approximate Nearest Neighbors*, ANN), já que o processo de recuperação de contexto relevante depende diretamente da qualidade e velocidade dessas buscas. Além disso, é essencial que o sistema ofereça suporte eficiente a consultas híbridas, ou seja, consultas que combinem filtros estruturados com similaridade vetorial. A latência também é um fator crítico, pois o tempo de resposta da recuperação impacta diretamente a experiência do usuário em sistemas interativos, como chatbots e assistentes inteligentes.

Outro aspecto importante é a capacidade do VDBMS de lidar com atualizações dinâmicas. Idealmente, ele deve permitir inserções e remoções de vetores de forma eficiente, ou ao menos fornecer mecanismos claros de reindexação periódica que evitem degradação do desempenho ao longo do tempo. A existência de indexadores de alto desempenho como HNSW[23] (*Hierarchical Navigable Small World*), que combina eficiência de busca com boa qualidade de resultados é fundamental.

6.3.3. Recuperação da informação

O objetivo central da Recuperação da Informação (RI) é realizar buscas, isto é, encontrar documentos relevantes com base em uma consulta fornecida pelo usuário. Um dos principais desafios dessa tarefa é que os termos utilizados na consulta nem sempre correspondem aos termos presentes nos documentos relevantes. Segundo [Moreira 2024], esse problema é conhecido como incompatibilidade de vocabulário (*vocabulary mismatch*), e decorre de dois fenômenos linguísticos frequentes: a sinonímia, quando diferentes palavras expressam o mesmo significado, e a polissemia, quando uma mesma palavra possui múltiplos significados.

Essa etapa do RAG pode utilizar diferentes métodos, desde sistemas clássicos de recuperação de informações como BM25 (Best Match 25) até o uso de modelos de linguagem especializados em representações vetoriais (modelos de embeddings) viabilizando a recuperação densa [Jurafsky and Martin 2025]. A principal diferença entre a recuperação usando o modelo BM25 e a recuperação densa é a forma como a informação é representada e em como a similaridade entre a pergunta e os fragmentos de texto é calculada.

6.3.3.1. BM25

O BM25 utiliza um algoritmo de recuperação de informação baseado na frequência de termos [Robertson et al. 2009]. Ele avalia a relevância de um documento para uma consulta com base na frequência com que os termos da consulta aparecem no documento. A

[21] https://www.elastic.co/

[22] https://opensearch.org/

[23] HNSW é um algoritmo baseado em grafos usado para busca aproximada de vizinhos mais próximos (ANN) em espaços de alta dimensão.

representação do texto é esparsa, assim cada termo é uma dimensão no vetor que representa um documento. O objetivo principal do BM25 é a correspondência exata de termos entre a consulta e os documentos e por isso não captura a semântica da linguagem.

Existem formas de minimizar esse problema expandindo os termos da consulta, mas isso depende da disponibilidade de grafos de conhecimento e dicionários especializados. Na Figura 6.9 estão representadas algumas dessas estratégias de expansão, como a utilizar de léxicos e até modelos de linguagem para gerar reformulações da consulta de forma a facilitar o sistema de recuperação em questão.

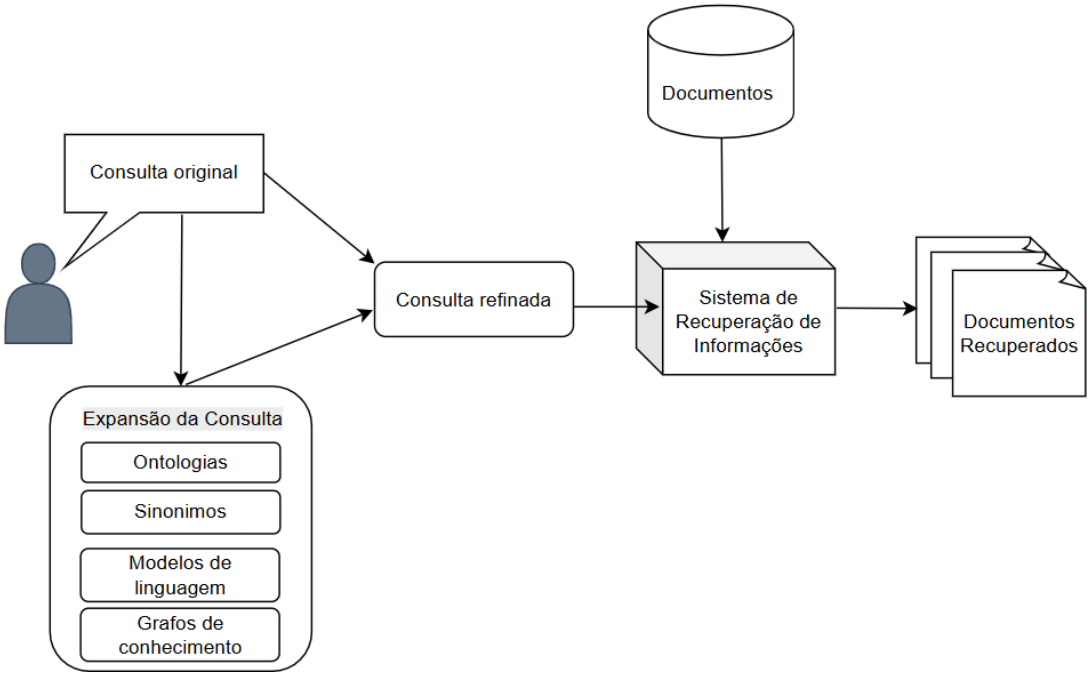

Figura 6.9. Fluxo para expansão de consultas

Outra maneira de expandir a consulta é denominada *Pseudo-Relevance Feedback*, onde o usuário faz uma consulta original, por exemplo "*asma crônica*", e o sistema executa BM25 e retorna os top-k documentos. A ideia baseia-se na suposição de que esses top-k documentos são relevantes — daí o termo "pseudo", já que não há uma verificação real com o usuário. A partir desses documentos, extraem-se os termos mais frequentes, como por exemplo: "*dispneia*", "*bronquite*", "*pulmão*" e "*tratamento*". Esses termos são combinados com a consulta original para formar uma nova consulta expandida: "*asma crônica bronquite pulmão tratamento*". Neste ponto, executa-se novamente o BM25, agora com a consulta expandida, e espera-se uma melhora na cobertura de sinônimos e contexto.

Como o BM25 utiliza diretamente os termos presentes nos documentos e nas consultas, o desempenho final do modelo depende da qualidade e relevância dessas palavras. Textos em geral apresentam muitos termos genéricos, flexões irrelevantes ou palavras não informativas (stop-words). Usar técnicas de pré-processamento do texto pode melhorar os resultados de forma significativa. As técnicas mais utilizadas são :

- Stemming: algoritmos [Orengo and Huyck 2001, Porter 1980] que visam unificar diferentes formas de uma mesma palavra ao reduzir seus sufixos, gerando uma representação comum. O principal benefício do stemming é aumentar a cobertura da busca, possibilitando a recuperação de um maior número de documentos potencialmente relevantes [Moreira 2024].

- Lematização: transforma uma palavra em sua forma canônica por exemplo: "correram"tem o lema "correr". Dessa forma, reduz-se o tamanho do vocabulário permitindo que variações de uma mesma palavra sejam reconhecidas como iguais. A qualidade do lematizador na língua em questão é fundamental para que se tenham bons resultados.

- Remoção de stop words: termos como “de”, “a”, “o”, “e”, etc. devem ser removidas por serem frequentes e pouco informativos. Mas em domínios específicos, há outras palavras altamente frequentes e genéricas (por exemplo: no domínio médico “paciente” e “sintoma”; em computação “usuário” e “sistema”). Assim, criar uma lista de stop words personalizada para cada corpus melhora a capacidade do BM25 de distinguir documentos relevantes.

- Normalização: se refere a remoção de acentos (diacríticos), padronização de caixa alta ou baixa. Esse processo visa garantir que termos semanticamente equivalentes não tratados de maneiras diferentes por variações ortográficas superficiais como versões em minúsculas e maiúsculas, ou com e sem acento.

6.3.3.2. Recuperação Densa

A busca lexical, como observado com BM25, se baseia na correspondência exata de termos entre a consulta e os documentos, ignorando sinônimos e diferentes grafias. Por outro lado, a busca semântica (ou recuperação densa) transforma a consulta em um vetor e localiza os documentos cujos vetores são similares nesse espaço. Na Figura 6.10 há uma apresentação simplificada dessa ideia usando duas dimensões: cada ponto verde é um vetor de um documento no corpus. No momento da busca, a consulta é incorporada no mesmo espaço vetorial (ponto vermelho) e os embeddings mais próximos são encontrados (documento relevante).

A recuperação densa utiliza modelos de embeddings para representar consultas e documentos como vetores densos em um espaço de embeddings. Esses vetores capturam a semântica da linguagem, permitindo que o modelo encontre documentos relevantes mesmo que não haja correspondência exata de termos. A representação dos dados é dita densa porque cada dimensão do vetor de números reais que representa o documento não corresponde a um termo individual, mas sim a características latentes aprendidas pelo modelo [Jurafsky and Martin 2025]. Assim, essa representação é mais rica semanticamente, mas pode ser mais custosa em termos de armazenamento e computação.

Na recuperação densa, compara-se a consulta e os fragmentos de texto por meio da similaridade semântica entre seus vetores. Para isso, utilizam-se diferentes métricas de similaridade, como a similaridade do cosseno formulada na Equação 2 (com a normaliza-

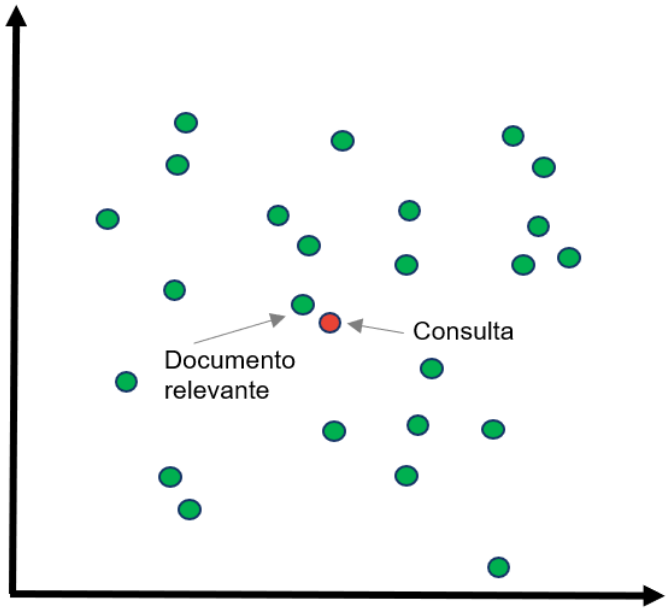

Figura 6.10. Recuperação densa

ção dos vetores pode ser transformada na métrica da distância do cosseno apresentada na Equação 3), a distância euclidiana na Equação 1, dentre outras [Eminagaoglu 2022].

$$d_{\text{euclidiana}}(a,b) = \|a-b\|_2 = \sqrt{\sum_{i=1}^{n}(a_i-b_i)^2} \tag{1}$$

$$\cos(\theta) = \frac{a \cdot b}{\|a\|\|b\|} = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2} \cdot \sqrt{\sum_{i=1}^{n} b_i^2}} \tag{2}$$

$$d_{\text{cosseno}}(a,b) = 1 - \cos(\theta) \tag{3}$$

6.3.3.3. Abordagens híbridas

A etapa de recuperação pode determinar o sucesso ou fracasso de um *pipeline* RAG. Se o *Retriever* não for capaz de localizar *chunks* relevantes, os resultados finais do RAG podem ser comprometidos e a chance do LLM gerador da resposta produzir alucinações aumenta, visto que o mesmo não receberá contexto adicional (aumentado) apropriado. Determinadas consultas podem ser melhor respondidas por técnicas de recuperação baseadas em palavras-chave, como o BM25. Enquanto outras apresentam melhor resultado com métodos de recuperação densos. Visando contornar as limitações individuais de cada abordagem, estratégias híbridas têm sido empregadas com bons resultados. Existem duas abordagens principais para construir um sistema de busca híbrido: fusão e reranking.

A **abordagem por fusão** combina os resultados obtidos por diferentes métodos de busca com base apenas nas pontuações fornecidas por cada um deles. Como essas

pontuações podem estar em escalas distintas, é comum aplicar algum tipo de normalização. Em seguida, uma fórmula agrega essas medidas de relevância para calcular uma pontuação final, que serve para reordenar os documentos [Qdrant 2025].

O *Reciprocal Rank Fusion* (RRF) [Cormack et al. 2009] é uma técnica de combinação de resultados que agrega listas de resultados ranqueadas retornadas por múltiplos sistemas de busca ou diferentes estratégias de recuperação. O objetivo é gerar uma lista final de resultados que seja superior a qualquer uma das listas individuais utilizadas. A ideia central é que documentos que aparecem no topo de várias listas têm alta probabilidade de serem relevantes de fato. Em vez de simplesmente somar ou ponderar as pontuações de relevância brutas dos diferentes sistemas (que podem estar em escalas diferentes e não ser diretamente comparáveis), a RRF considera a posição (rank) do documento em cada lista.

Na Tabela 6.6, observa-se que a abordagem híbrida com RRF apresenta desempenho superior em diversos conjuntos de dados, superando os métodos BM25, o denso e o híbrido convencional. A métrica utilizada na comparação é o nDCG@10, do inglês *Normalized Discounted Cumulative Gain* [Järvelin and Kekäläinen 2002], uma das principais métricas de RI para avaliar a qualidade de rankings gerados por sistemas de busca. Ela considera tanto a relevância dos documentos quanto sua posição no ranking, atribuindo maior peso aos documentos mais relevantes posicionados nas primeiras colocações. O valor do nDCG é normalizado entre 0 e 1, sendo 1.0 o valor correspondente ao ranking ideal.

Tabela 6.6. Compara as pontuações do NDCG@10 com diferentes métodos de busca. Adaptado de [Bogan et al. 2025]

Dataset	BM25	Denso	Híbrido	Híbrido com RRF	Diferença percentual
NFCorpus	0,3065	0,2174	0,3076	0,2977	3,22%
ArguAna	0,4258	0,4239	0,4507	0,4476	0,69%
FIQA	0,2389	0,2004	0,2693	0,2474	8,13%
Trec-Covid	0,6087	0,2718	0,5905	0,5877	0,47%
SciDocs	0,155	0,1075	0,1602	0,1525	4,81%
Quora	0,7424	0,8256	0,8452	0,796	5,82%

A **abordagem por *reranking*** consiste na aplicação de modelos computacionalmente mais sofisticados e custosos — como modelos densos do tipo *bi-encoder* (por exemplo, Contriever [Izacard et al. 2021] ou E5 [Wang et al. 2022]) ou ainda *cross-encoders* como *ms-marco-MiniLM-L4-v2*[24] — sobre um subconjunto reduzido de documentos previamente recuperados. Inicialmente, utiliza-se um método de recuperação eficiente, como o BM25, para selecionar os top-N documentos mais relevantes com base em critérios lexicais. Em seguida, esse conjunto é reordenado por um modelo de *reranking* que realiza uma avaliação mais precisa da relevância, permitindo selecionar os top-k resultados finais ($k << N$). Essa estratégia em duas etapas equilibra desempenho computacional e precisão na recuperação da informação.

[24]https://huggingface.co/cross-encoder/ms-marco-MiniLM-L4-v2

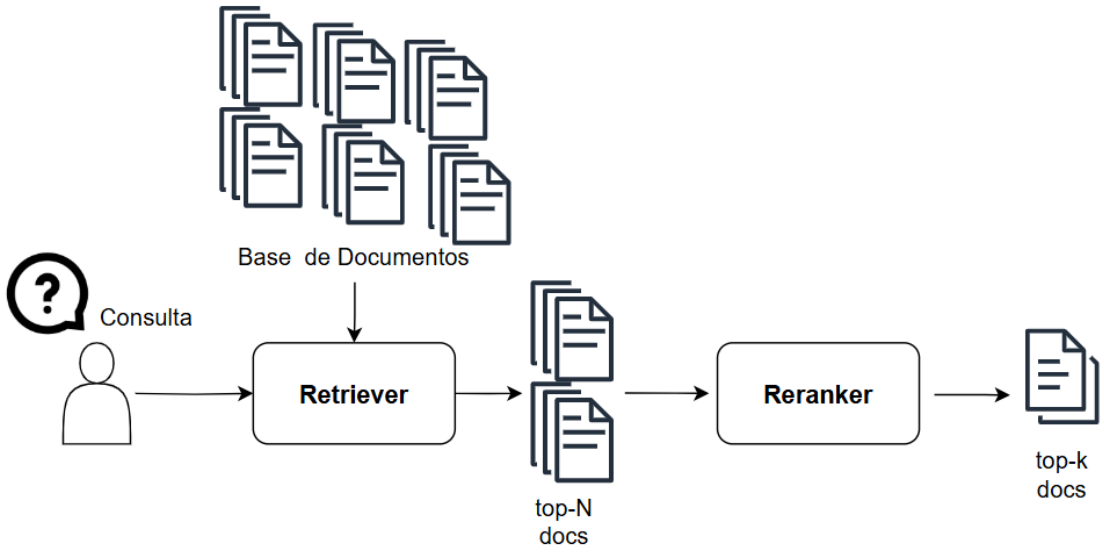

Figura 6.11. Sistema de Recuperação usando *Reranking*

6.3.4. Geração da resposta

Nesta seção, serão abordados os processos e componentes do *Generator* apresentado na Figura 6.7. Essa etapa é fundamental no RAG pois é onde o que foi recuperado pelo *Retriever* é utilizado para gerar a resposta. Este processo envolve várias técnicas que devem ser cuidadosamente avaliadas e ajustadas para o caso de uso em questão: qual LLM será utilizado, qual a melhor estratégia de decodificação na geração do texto, qual o *prompt* ideal e como usar o contexto recuperado da melhor forma a evitar confundir o LLM e minimizar alucinações.

6.3.4.1. Modelos de linguagem

Os LLMs revolucionaram o processamento de linguagem natural graças a avanços em arquiteturas baseadas em *deep learning*. A arquitetura transformer [Vaswani et al. 2017] trouxe como grande benefício a capacidade de processar sequências inteiras de dados (como textos) simultaneamente, por meio do mecanismo de autoatenção. Isso permitiu o processamento paralelo que acelera significativamente o treinamento destes modelos e beneficia a escalabilidade, pois facilita o treinamento de modelos maiores com mais dados. Também melhora muito a captura de dependências, ou seja, o modelo consegue identificar relações entre elementos distantes dentro da sequência com mais eficiência. Abaixo um exemplo destacado em [Paes et al. 2024]:

"A garota de blusa amarela com uma frase em que os verbos estavam em letras pretas, que andava tão rápido e nunca em linha reta, a ponto de passar pelas nossas vistas como se fosse quase um furacão, tinha na parte de trás da sua blusa uma frase atribuída a Gandhi: "Acreditar em algo e não vivê-lo, é desonesto".

Conforme destacam os autores do trecho, se quiséssemos descobrir a cor das letras

Tabela 6.7. Principais LLMs na área médica. Adaptado de [Carchiolo and Malgeri 2025]

Modelo	Objetivo	Características
BioGPT [Lee et al. 2020]	PLN biomédico	Treinado com literatura biomédica para geração de texto e resposta a perguntas clínicas
GatorTron [Yang et al. 2022]	Análise de dados clínicos	Treinado com milhões de prontuários eletrônicos para melhorar a compreensão da linguagem clínica
MedPaLM [Singhal et al. 2025]	Diagnóstico e perguntas clínicas	Combina o PaLM com dados médicos para responder a perguntas clínicas
PubMedBERT [Gu et al. 2021]	PLN biomédico	Versão do BERT otimizada para artigos do PubMed, útil para classificação de texto e extração de informações
Galactica [Taylor et al. 2022]	Pesquisa científica	Treinado em textos científicos, incluindo publicações médicas
ClinicalBERT [Alsentzer et al. 2019]	Compreensão da linguagem clínica	Adaptação do BERT para registros eletrônicos de saúde (EHR)
BioBERT [Lee et al. 2020]	Biologia e aplicações médicas	Treinado com dados do PubMed e PMC para aprimorar a precisão do PLN na saúde

com que a palavra "*Acreditar*" foi escrita, precisaríamos associá-la à palavra "*verbo*" e, então, retornar ao início da frase para notar que os verbos estão em preto. Além disso, há uma grande quantidade de palavras entre a informação sobre a cor e o primeiro verbo da frase de Gandhi. Uma rede recorrente (RNN), usada para esse tipo de problema antes do advento dos transformers, tem mais dificuldade em aprender essas conexões.

O Transformer é composto por duas grandes partes: o *encoder*, responsável por codificar o texto e gerar representações linguísticas, e o *decoder*, encarregado de decodificar essas representações e gerar texto. Modelos como GPT e LLaMA, são compostos por *decoders* e são otimizados para geração sequencial de texto — ou seja, predizem a próxima palavra com base nas anteriores. Por outro lado, arquiteturas bidirecionais, como o BERT e os modelos de embeddings especializados em gerar representações vetoriais do texto, tratados na Seção 6.3.2, utilizam *encoders* e são voltadas para compreensão contextual. Modelos BERT predizem palavras mascaradas considerando o contexto à esquerda e à direita e são amplamente utilizados em tarefas como classificação. Já os modelos *encoder-decoder*, como T5 e BART, combinam as capacidades de codificação e geração, sendo usados para tarefas como tradução automática e sumarização.

Para RAG os modelos precisam ter autorregressivos pois são usados na geração da resposta final. Mais detalhes sobre a geração do texto na Seção 6.3.4.2. Dentro da área médica alguns dos principais LLMs, seus objetivos e características estão listados na Tabela 6.7.

6.3.4.2. Geração de texto

A geração de texto refere-se à produção de sequências textuais condicionadas a uma entrada textual. Essa capacidade é empregada em diversas tarefas de *PLN*, como sumarização, tradução, geração de texto em diálogo aberto, transcrição de fala para texto, conversão de imagens em descrições textuais, entre outras. Segundo [Jurafsky and Martin 2025], a utilização de modelos de linguagem para gerar texto representa uma das áreas de maior impacto dos modelos neurais no campo do *PLN*. A geração de texto, juntamente com a criação de imagens e a geração de código, compõe uma nova vertente da Inteligência Artificial (IA), frequentemente denominada **IA generativa**.

Os modelos de linguagem autorregressivos usados na geração automática de texto, seguem o princípio da **Modelagem de Linguagem Causal** (em inglês, *Casual Language Modeling*) , no qual cada palavra gerada depende do histórico das palavras anteriores. A geração ocorre de forma sequencial: a cada etapa, o token produzido é acrescentado ao contexto e utilizado como base para prever o próximo elemento da sequência.

Os modelos são treinados com grandes volumes de texto (livros, artigos, websites, etc.) para prever a próxima palavra/token em uma sequência. O modelo gera tokens um por um, escolhendo as palavras seguintes com base nas probabilidades aprendidas. Entrando a sequência "O céu está muito ___." o modelo provavelmente responderá com palavras como "azul" ou "nublado", pois são respostas prováveis. Assim, quando um modelo como o GPT recebe uma sequência de entrada, ele calcula a distribuição de probabilidade para o próximo token. Chamamos de **Estratégia de Decodificação** o mecanismo que transforma essa distribuição em uma escolha concreta de palavras, determinando como a sequência será construída.

Diferentes estratégias de decodificação podem levar a resultados significativamente diferentes em relação à qualidade, coerência e diversidade do texto. Algumas das estratégias de decodificação mais comumente usadas são as seguintes:

- Busca Gulosa: seleciona a palavra com a maior probabilidade como a próxima palavra. É o método mais simples, mas computacionalmente muito eficiente. Oferece baixa diversidade, levando a repetições [Jurafsky and Martin 2025].

- Busca por Feixes [Freitag and Al-Onaizan 2017]: explora múltiplas alternativas por etapa e seleciona a de maior probabilidade total, mas consome muitos recursos e pode gerar saídas repetitivas.

- Amostragem Top-k [Fan et al. 2018]: o modelo restringe a escolha da próxima palavra às k mais prováveis e realiza uma seleção aleatória entre elas, conforme a distribuição de probabilidade. Essa limitação introduz variabilidade e criatividade no texto gerado, mas pode afetar a consistência semântica quando k é excessivamente alto.

- Amostragem Top-p [Holtzman et al. 2019]: escolhe o menor conjunto possível de palavras cuja probabilidade cumulativa excede a probabilidade p. A massa de probabilidade é então redistribuída entre esse conjunto de palavras. O valor de p con-

trola a diversidade do texto gerado. Valores mais altos de p levam a um texto mais diverso, enquanto valores mais baixos geram um texto mais previsível.

- Temperatura: reduzir a temperatura do softmax significa tornar a distribuição de probabilidade mais nítida [Kamath et al. 2019]. A temperatura é um hiperparâmetro que controla a aleatoriedade do processo de decodificação no LLM. Valores mais baixos resultam em texto mais previsível e repetitivo, e o modelo se torna determinístico, enquanto uma temperatura mais alta gera texto mais diverso e criativo.

A escolha de uma estratégia de decodificação deve considerar um equilíbrio entre diversidade, coerência e eficiência computacional. Em muitos casos, uma combinação dessas estratégias ou modificações personalizadas podem ser usadas.

6.3.4.3. Organização dos *prompts*

A aprendizagem em contexto (*In-Context Learning*, ICL), mencionada na Seção 6.3, refere-se à capacidade dos LLMs de realizar novas tarefas com base apenas nos exemplos e instruções fornecidos na entrada, sem necessidade de atualizar seus parâmetros durante a inferência. Nesse paradigma, o modelo não passa por nenhum processo de treinamento adicional, ou seja, não há atualização de gradientes, mas sim utiliza o contexto do *prompt* para simular o comportamento esperado. Assim, o LLM é capaz de se adaptar a tarefas ou conceitos que não foram explicitamente vistos durante o treinamento, generalizando a partir de poucos exemplos apresentados no momento da consulta (inferência).

A ICL pode ocorrer em um cenário onde não se fornece nenhum exemplo (*zero-shot*), ou seja, apenas a solicitação (a tarefa a ser resolvida) é enviada ao modelo com a instância de teste para a qual o modelo deve fazer gerar a resposta. Também é possível utilizar a abordagem com poucos exemplos (*few-shot*) onde utiliza-se um conjunto de demonstrações na tarefa-alvo, consistindo na entrada e na saída desejada. Portanto, além de enviar a descrição da tarefa, enviamos as demonstrações, seguidas de um único exemplo não rotulado para o qual a previsão deverá ser feita pelo LLM. A ideia é que o modelo utilize esses exemplos para fornecer a resposta à tarefa para a qual não foi especificamente treinado. Toda essa entrada é chamada de "prompt".

Assim, no contexto dos LLMs, um *prompt* é um enunciado em linguagem natural que orienta o modelo sobre qual tarefa executar. Ele pode assumir diferentes formas, como uma pergunta direta, uma afirmação contextual, uma instrução descritiva ou até uma solicitação de tarefa específica, como resumir ou classificar um conteúdo [Paes et al. 2024].

A engenharia de *prompts* é a prática de projetar e refinar *prompts* de entrada para aprimorar as respostas que obtemos do modelo de linguagem. É uma atividade empírica. As saídas do modelo variam de acordo com o *prompt* utilizado, os exemplos selecionados e a ordem dos exemplos. Embora os *prompts* possam ser redigidos de maneira flexível, os LLMs são altamente sensíveis à sua formulação. A Figura 6.12 mostra um processo de criação manual de *prompts*. A segunda etapa na Figura (criação do *prompt*) pode ser automatizada e/ou usar fontes externas para complementar o *prompt* e fornecer um contexto mais relevante para melhorar a resposta do LLM.

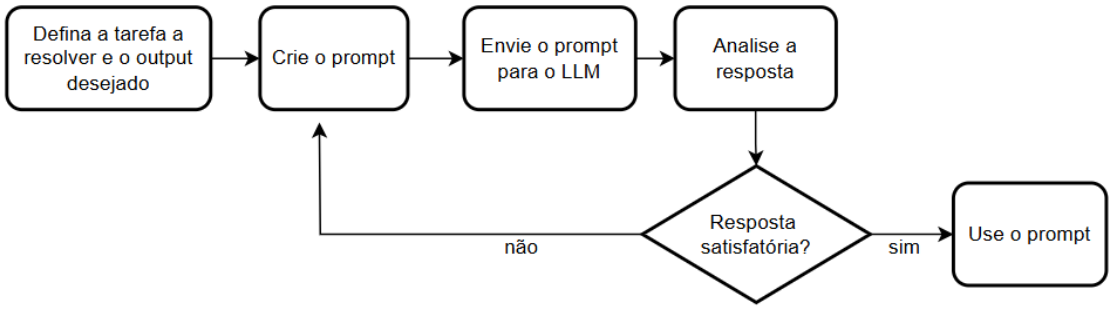

Figura 6.12. Ciclo de Engenharia de *Prompts*

Diversas estratégias de criação de *prompts* foram desenvolvidas. Uma das mais utilizadas é a Cadeia de Pensamento (*Chain-of-Thought*, CoT) [Wei et al. 2022]. Ela aprimora a capacidade de realizar raciocínios complexos, incorporando etapas intermediárias. É uma estratégia que faz o modelo explicitar seu raciocínio antes de chegar à resposta final. Em vez de pedir uma resposta direta, o *prompt* instrui o modelo a "pensar passo a passo".

[Wu et al. 2024] desenvolveram um método baseado CoT para detecção e correção de erros médicos em notas clínicas, desenvolvido para o desafio MEDIQA-CORR 2024. Os autores identificam manualmente três tipos comuns de erro médico: diagnóstico, intervenção e manejo. Na Figura 6.13 é possível visualizar os três *prompts* distintos para cada categoria de erro. O modelo GPT-4 é guiado por etapas: primeiro detecta se há erro; caso afirmativo, identifica a sentença; por fim, gera a correção.

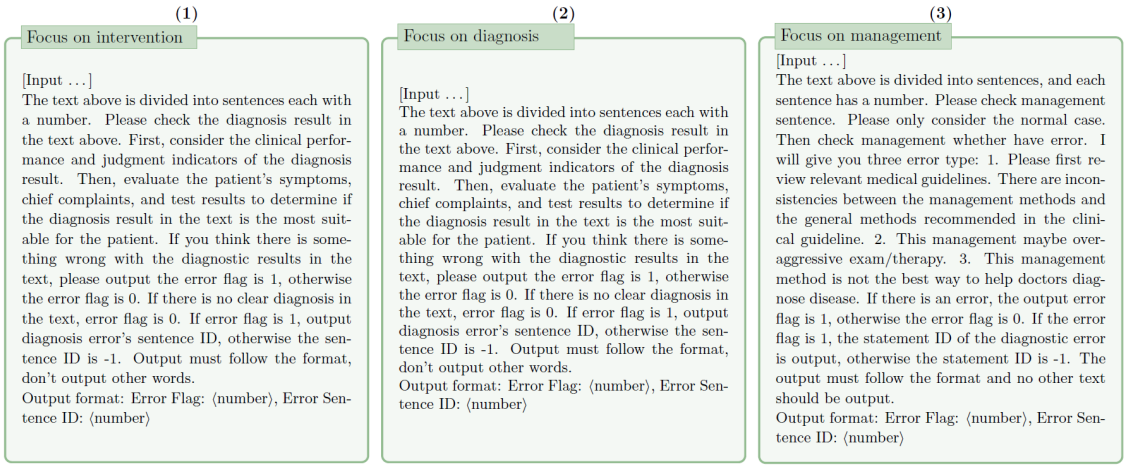

Figura 6.13. Três tipos de *prompts* de Cadeia de Pensamento (CoT) utilizados para identificar erros de intervenção, diagnóstico e gestão, respectivamente. Extraído de [Wu et al. 2024]

Uma estratégia eficaz na engenharia de *prompts* consiste em decompor tarefas complexas em subtarefas menores. Cada uma dessas subtarefas é apresentada ao LLM por meio de um*prompt* específico, cuja saída pode ser reutilizada como entrada em uma etapa posterior. Esse processo é conhecido como encadeamento de *prompts* (*prompt chaining*), no qual a resolução de uma tarefa ocorre de forma progressiva, por meio de uma

sequência estruturada de instruções. Essa abordagem se destaca em casos onde o modelo teria dificuldade em lidar com uma instrução muito extensa ou complexa de uma só vez. Ao aplicar o encadeamento, cada resposta intermediária pode ser refinada, complementada ou transformada por novos *prompts*, até que se alcance o resultado final esperado.

Reflexion [Shinn et al. 2023] é uma técnica de *prompting* composta, iterativa e reflexiva. Ela se enquadra em um tipo mais avançado de engenharia de *prompt*, onde o LLM é induzido a aprender com seus próprios erros por meio de autoavaliação e reutilização estratégica de contexto textual. *Reflexion* opera inteiramente via linguagem natural. Ele não treina o modelo, mas sim estrutura a interação com o modelo de forma estratégica, por meio de *prompts* que executam a tarefa, que avaliam o próprio desempenho, que integram a memória das reflexões passadas como contexto para a próxima tentativa.

No contexto do RAG, a escolha das estratégias de *prompting* dependem do domínio, mas também do LLM, alguns modelos tem uma capacidade maior de seguir instruções, por exemplo. Os *chunks* recuperados pelo *Retriever* são ordenados segundo a métrica de similaridade usada na recuperação e seleciona-se os top-k com melhores valores nesta métrica. A escolha do k a utilizar é um ponto importante e para isso deve-se considerar a janela de contexto do modelo e o tamanho máximo do *chunks* (ver discussão na Seção 6.3.2) para evitar o estouro do limite de entrada do LLM.

O número de top-k documentos a considerar na etapa de recuperação de um sistema RAG é uma decisão crítica. Não existe um único valor ótimo de top-k, mas sim recomendações baseadas no tipo de aplicação, modelo usado e tolerância a erro. Valores baixos de k (de 3 a 5) são recomendados para perguntas factuais ou quando se sabe que a base de dados tem alta qualidade. Usar $k > 10$ aumenta a diversidade e cobertura de possíveis evidências, mas eleva o risco de conflito (informações contraditórias) e exige mecanismos de fusão ou seleção. Para *chunks* longos um k menor é recomendado.

Os *chunks* são recebidos pelo *Generator* e serão organizados e adicionados ao *prompt* que será enviado ao LLM. Neste ponto, as técnicas mencionadas e outras mais de engenharia de *prompt* podem ser aplicadas e incluindo variações dependendo do caso de uso do RAG, como por exemplo sumarização dos *chunks* para casos em que sejam muitos a considerar ou quando o LLM utilizando possui uma janela de contexto pequena. Na Figura 6.14 um exemplo de um *prompt* simples e genérico para RAG com $k = 3$.

6.3.4.4. Fine-tuning de modelos

Em uma arquitetura RAG, a princípio, não é necessário fazer fine-tuning do LLM para gerar as respostas, mas em contextos mais especializados ou críticos como saúde, o fine-tuning pode melhorar significativamente a performance. O fine-tuning de modelos de linguagem (LLMs) abrange diversas técnicas para adaptar um modelo pré-treinado a tarefas ou domínios específicos.

O termo fine-tuning pode se referir à extensão do ajuste de parâmetros (quanto do modelo será treinado) ou ao objetivo do treinamento (o que o modelo aprende). Quanto à extensão do ajuste ele pode ser total ou parcial. O ajuste total atualiza todos os parâmetros do modelo, é computacionalmente custoso e apresenta o risco do esquecimento


```
Baseando-se exclusivamente nos documentos fornecidos abaixo,
responda à pergunta a seguir.

Se a informação solicitada não estiver presente, diga "As
informações fornecidas não são suficientes para responder com
segurança."

Inclua na resposta trechos que justifiquem suas afirmações.


Pergunta: [Pergunta]


Documentos:

[Chunk 1] ...

[Chunk 2] ...

[Chunk 3] ...
```

Figura 6.14. Exemplo de *prompt* para RAG

catastrófico (*catastrophic forgetting*) [Li et al. 2025], que se refere ao esquecimento do conhecimento prévio. Ajustes parciais são mais leves, exigindo menos memória e treino, por exemplo congelar camadas iniciais (que capturam características gerais) e ajustar apenas as camadas superiores. PEFT (*Parameter-Efficient Fine-Tuning*) [Ding et al. 2023] é um conjunto de técnicas projetadas para ajustar modelos de linguagem grandes de forma eficiente, modificando apenas uma pequena fração dos parâmetros do modelo original. Seu objetivo é reduzir custos computacionais, minimizando o risco de *catastrophic forgetting*. Técnicas como LoRA (*Low-Rank Adapter*) [Hu et al. 2022] e QLoRA (*quantized Low-Rank Adapter*) [Dettmers et al. 2023] congelam os pesos originais do modelo base e adicionam pequenas matrizes de baixo posto (*low-rank*) treináveis, que capturam os ajustes necessários para a tarefa.

Quanto ao objetivo, o *fine-tuning* pode adotar diferentes estratégias, dependendo da finalidade da adaptação do modelo. O ajuste supervisionado (*Supervised Fine-Tuning*, SFT) consiste no treinamento do modelo em um conjunto de dados anotado para tarefas específicas, como classificação. No ajuste por instrução (*Instruction Tuning*) o modelo é treinado para seguir instruções explícitas em linguagem natural, com o objetivo de torná-lo mais útil e alinhado a comandos humanos [Ouyang et al. 2022]. Neste caso, utiliza-se pares instrução–resposta para melhorar a capacidade do modelo de seguir comandos diversos. O método de aprendizado por reforço a partir do feedback humano (*Reinforcement Learning from Human Feedback, RLHF)* [Ouyang et al. 2022], ajusta o comportamento

do modelo com base em preferências humanas: as saídas do modelo são ranqueadas por humanos. O ChatGPT, por exemplo, foi ajustado com RLHF. Nessa linha, a Otimização Direta por Preferência (*Direct Preference Optimization, DPO*) [Rafailov et al. 2023] é uma alternativa ao RLHF, pois treina diretamente com base em pares de preferência (resposta preferida, não preferida), otimizando o modelo diretamente para gerar respostas mais alinhadas ao julgamento humano.

O RAFT (*Retrieval-Augmented Fine-Tuning*) [Zhang et al. 2024] é uma técnica de ajuste supervisionado que especializa um LLM para funcionar melhor dentro de um sistema RAG. Os autores comparam o desempenho de LLMs com o comportamento de estudantes em provas com consulta: o RAG tradicional é como um aluno que só olha o livro na hora da prova, sem ter estudado previamente; o RAFT é um aluno que estudou os livros antes da prova, ou seja, o modelo treina com os documentos antes de ser testado com recuperação.

Em vez de usar um LLM genérico e simplesmente adicionar os documentos recuperados no *prompt*, como no RAG tradicional, o RAFT treina o modelo com exemplos explícitos de como raciocinar com documentos relevantes e ignorar os irrelevantes. Os autores usam o modelo LLaMA 2 de 7 bilhões de parâmetros e criam um conjunto de dados sintético onde cada exemplo tem os seguintes elementos:

- Pergunta
- Conjunto de documentos recuperados que contém documentos relevantes (contêm a resposta) mas também documentos irrelevantes (distratores, que devem ser ignorados).
- Resposta correta gerada a partir dos documentos (A resposta usa conteúdo apenas dos documentos relevantes).
- Explicação com raciocínio (*Chain-of-Thought*): Inclui citações textuais dos documentos relevantes para justificar a resposta.

Depois de treinado o modelo é usado dentro do *pipeline* do RAG normalmente. Dessa forma, o objetivo do RAFT é ensinar o LLM usado no RAG a raciocinar com base em múltiplos documentos, ignorar informações irrelevantes e usar informações relevantes de forma correta.

Na mesma linha, [Jeong et al. 2024] apresentam o framework Self-BioRAG de ajuste do LLM usado no RAG visando melhorar a capacidade em responder perguntas médicas. Para isso, os autores realizam fine-tuning em dois modelos LLaMA2. Primeiro, fazem ajuste do modelo crítico *C* que aprende a prever quatro tokens reflexivos: (i) identificar se uma questão requer recuperação (RET); (ii) determinar se a evidência recuperada fornece informações úteis para resolver uma questão (REL); (iii) avaliar se todas as declarações de respostas podem ser apoiadas por evidências (SUP); (iv) avaliar se todas as declarações de respostas são uma resposta útil à questão (USE). O modelo *C* é usado para anotar um dataset maior com esses tokens reflexivos. Esse dataset é então usado para ajustar o LLM gerador.

O LLM ajustado é o modelo usado em produção, e ele aprendeu, com base nos exemplos anotados pelo modelo crítico *C*, a decidir se precisa recuperar ou não, a avaliar relevância e fundamentar a resposta. Na Figura 6.15 podemos ver a comparação de três abordagens para responder perguntas clínicas. O esquema C demonstra o LLM gerador atuando primeiramente na verificação da necessidade de recuperar contexto. Caso não seja necessário o LLM responde diretamente. Sendo necessário ele segue o fluxo de uma aplicação RAF e o contexto é recuperado da base PubMed Central[25] (PMC). Depois de recuperado o contexto é avaliado pelo LLM que seleciona a melhor evidencia a ser usada na geração da resposta.

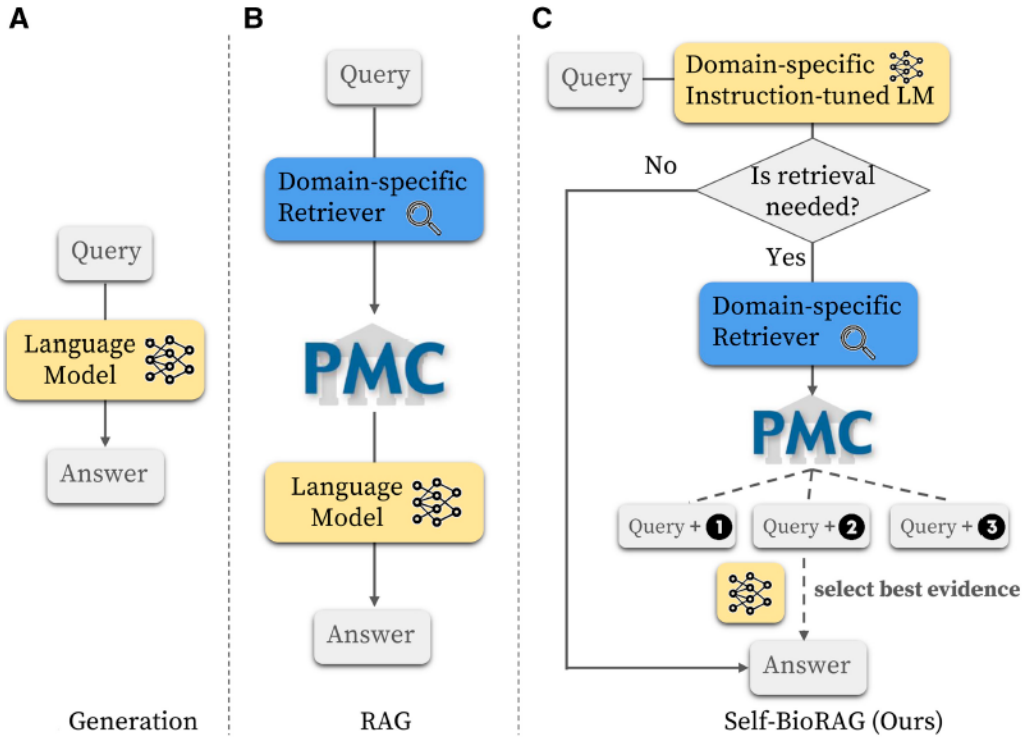

Figura 6.15. Comparação entre três estruturas para gerar respostas a perguntas: A) usando somente LLM, B) geração aumentada de recuperação (RAG) e C) Usando modelo ajustado Self-BioRAG. Extraído de [Jeong et al. 2024]

6.4. Avaliação dos resultados do RAG

Avaliar sistemas RAG apresenta desafios devido à sua estrutura híbrida que envolve a participação de múltiplos componentes. De maneira geral, o desempenho do *pipeline* RAG pode ser avaliado examinando os dois componentes principais: *Retriever* e *Generator*.

Esforços foram feitos para avaliar RAG no contexto clínico com desenvolvimento de benchmarks de desempenho que utilizam respostas de múltipla escolha ou verdades categóricas para respostas como MedRAG [Xiong et al. 2024]. Entretanto esse tipo de avaliação falha em capturar as complexidades e riscos associados com gerações de respostas abertas [Chowdhury et al. 2025]. Para avaliar RAG é necessário dispor de métricas

[25]https://pmc.ncbi.nlm.nih.gov/tools/textmining/

capazes de aferir tanto o desempenho global do sistema quanto o funcionamento individual de cada módulo de forma a diagnosticar as origens dos erros e compreender como eles se manifestam ao longo do processo [Ru et al. 2024].

6.4.1. Métricas tradicionais

No caso *Retriever*, métricas traicionais utilizadas na tarefa de Recuperação de Informação (RI) clássica como recall@k e MMR [Moreira 2024], entre outras, podem ser utilizadas. O problema que essas métricas dependem de esquemas fixos de segmentação (chunking) e anotações específicas para RI considerando estes esquemas, isso significa que variar o esquema de *chunking* compromete a anotação inicial que não é necessariamente mapeável para um novo esquema.

Já para o *Generator*, métricas usadas na geração de texto poderiam ser usadas desde que exista um texto de referência. Essas métricas comparam o texto gerado à referência. Por exemplo, em perguntas e respostas cada pergunta tem uma resposta esperada que será comparada à resposta gerada.

Métricas como BLEU e ROUGE são amplamente usadas na geração de texto. Elas são métricas de sobreposição de N-gramas[26], mas apresentam diversas limitações na avaliação de textos gerados por LLM, que é o caso de sistemas RAG. Estas métricas se baseiam na correspondência exatas de palavras ou frases, ignorando sinônimos, paráfrases e expressões semanticamente equivalentes, mas lexicalmente diferentes. Por exemplo, o texto gerado "*O felino descansou no carpete*"ao ser avaliado contra o texto de referência "*O gato sentou no tapete*"apresentará pontuação baixa mesmo que o significado esteja correto. Enquanto que o texto gerado "*O cachorro preguiçoso salta sobre a rápida raposa marrom*"avaliado contra a referência "*A rápida raposa marrom salta sobre o cachorro preguiçoso*"poderia obter uma pontuação enganosamente alta com base na sobreposição de palavras individuais (unigramas), mascarando a mudança fundamental de significado devido à ordem das palavras.

BERTScore [Zhang et al. 2020] é uma métrica que também pode ser usada na geração de texto. Ela é baseada em embeddings de modelos *encoder-only* como o BERT e tem mais sucesso em lidar com casos semanticamente equivalentes mas lexicalmente diferentes. Entretanto, em exemplos de negação ou de contradição onde pode atribuir alta similaridade, porque não é sensível a negação de forma explícita. No caso do texto gerado "*Tomar aspirina é excelente para pressão alta*"comparado ao texto de referência "*Tomar aspirina é um risco para pressão alta*"pode resultar em alta similaridade, pois as duas frases compartilham quase o mesmo vocabulário apesar de terem sentidos opostos. O BERTScore não entende negação nem oposição lógica explícita. Isso significa que, nesse caso, uma resposta perigosa (como recomendar aspirina indevidamente) pode ser avaliada pelo BERTScore como "boa". Além disso, há a questão do conhecimento do modelo usado no BERTScore sobre o domínio: utilizar modelos que não reconhecem o domínio da saúde pode não identificar a similaridade de exemplos como este: "*A pressão arterial elevada pode aumentar o risco de AVC*"e "*Hipertensão eleva a probabilidade de um derrame cerebral*".

[26]Um n-grama é uma sequência contígua de n itens de uma determinada amostra de texto ou fala

Em textos longos, o BERTScore perde precisão e interpretabilidade. Além disso, modelos BERT-base têm limite de 512 tokens em média. Textos maiores são truncados, o que pode fazer com que partes relevantes da resposta sejam ignoradas. Mesmo para textos longos menores que 512 tokens, BERTScore atribui peso igual para todos os tokens no calculo da similaridade. Não distingue entre um token irrelevante e um que expressa uma afirmação crítica (por exemplo: "não", "risco", "fatal"). Consequentemente, em textos longos, afirmações importantes podem se diluir no meio de conteúdo superficial. Por isso, em tarefas como RAG, é preferível complementar BERTScore com com métricas baseadas em inferência textual (*entailment*), que indicam contradição, implicação ou neutralidade entre duas sentenças. Para isso é necessário modelos de inferência textual também especializados no domínio desejado.

6.4.2. LLMs como avaliadores

Um um conjunto com perguntas e respostas de referência no domínio de interesse, muitas vezes não está disponível para averiguar se as configurações do sistema RAG projetado alcançam bons resultados. Além disso, a construção destes conjuntos é bastante custosa exigindo tempo considerável dos anotadores (humanos que avaliam os dados), que algumas vezes precisam ser especialistas no domínio. Adicionalmente, a avaliação do RAG precisa conectar os resultados da geração e da recuperação e isso não é simples de anotar. Por exemplo, até que ponto a resposta gerada se baseia efetivamente nos trechos recuperados? há informações incluídas que não estão presentes nos top-k documentos retornados?

Existe um movimento de avaliar sistemas RAG usando LLMs que é aderente ao que ocorre para uma série de tarefas de IA que usam esses modelos como julgadores ou avaliadores. Essas iniciativas são classificadas sob a descrição "LLM as a judge" [Zheng et al. 2023]. São técnicas baseadas no paradigma do aprendizado em contexto (*In-Context Learning*) popularizado a partir do modelo GPT-3 [Brown et al. 2020].

Frameworks de avaliação RAG apresentam formas de avaliar automaticamente os resultados usando métricas que não exigem uma base anotada, como por exemplo RAGAS [Es et al. 2024] e RAGTriad [Trulens 2024] . Eles utilizam LLMs para avaliar a resposta geral, mas também buscam avaliar os resultados dos diversos componentes da arquitetura RAG. Estão integrados a outros frameworks para desenvolvimento de soluções RAG e agentes como LangChain[27] e Llama Index[28].

Recentemente, o framework de avaliação RAGChecker [Ru et al. 2024] apresentou uma ampla consolidação das métricas e seu principal diferencial em relação ao RAGAS é o de considerar para o cálculo das métricas verificações detalhadas no nível de afirmações dentro da resposta e dentro dos fragmentos. Os autores também destacam que as métricas são projetadas para fornecer *insights* claros sobre as fontes de erros dentro dos diversos componentes RAG. Muitas destas métricas não exigem uma base anotada e usam LLM como julgadores, que tem sido uma tendência em várias tarefas, em especial em recuperação de informações [Bencke et al. 2024].

A RAG Triad, ilustrada na Figura 6.16, é uma proposta de avaliação automática de

[27]https://www.langchain.com/
[28]https://www.llamaindex.ai/

sistemas RAG implementada na ferramenta TrueLens[29]. Essa plataforma de software foi desenvolvida com o objetivo de mensurar a qualidade e a eficácia de aplicações baseadas em LLMs por meio de funções de feedback. Essas funções avaliam a qualidade de entradas, saídas e resultados intermediários em diferentes contextos de uso, como sistemas de perguntas e respostas, sumarização, RAG e agentes inteligentes. As três dimensões da RAG Triad são avaliadas por *prompts* que solicitam notas em uma escala de 0 a 10, adotando a estratégia *Chain-of-Thought* (CoT) [Wei et al. 2022] para fundamentar as respostas.

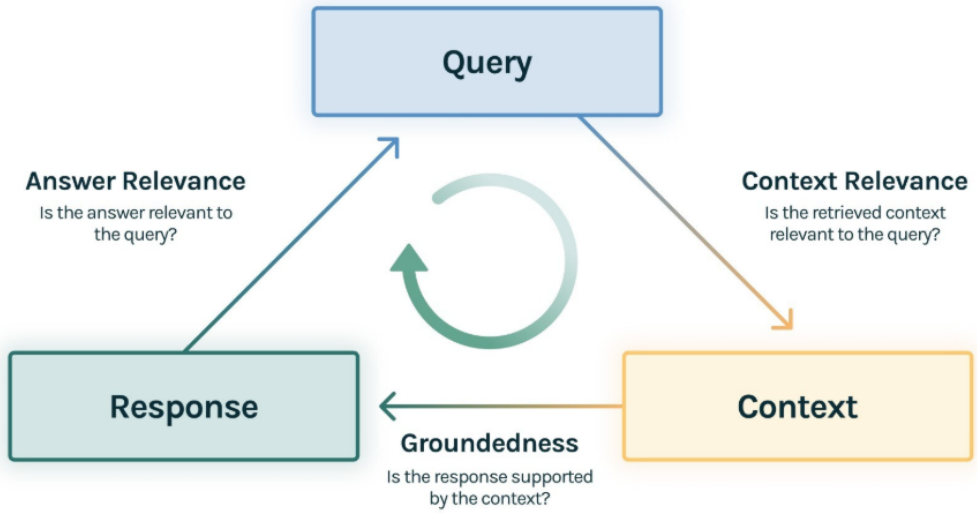

Figura 6.16. As três dimensões da avaliação RAG Triad. Extraído de [Trulens 2024]

O framework RAGAS proposto por [Es et al. 2024], implementa um conjunto de métricas para avaliar diferentes dimensões da resposta sem precisar de anotações humanas. Três métricas principais estão descritas a seguir.

Fidelidade (*Faithfulness*): a resposta deve estar fundamentada no contexto informado (*chunks* recuperados pelo *Retriever*). Isso é importante para evitar alucinações e garantir que o contexto recuperado possa servir de justificativa para a resposta gerada. Para esta métrica RAGAS usa LLM em dois passos: (1) envia a resposta gerada solicitando a extração de um conjunto de declarações (*claims*), 2) envia as declarações obtidas em (1) e pede que o LLM determine quais delas estão fundamentada pelas informações presentes no contexto inicial (conjuntos de *chunks* recuperados) e explique o porque de sua conclusão. O resultado final da Equação 4 corresponde ao percentual de declarações que estão fundamentadas pelo contexto de *chunks*. Quanto maior a fidelidade maior a aderência da resposta final ao contexto informado (*chunks*).

$$\text{Faithfulness} = \frac{\text{Nro de declarações na resposta gerada que podem ser inferidas do contexto}}{\text{Número total de declarações na resposta gerada}} \tag{4}$$

[29] https://www.trulens.org/

Relevância da resposta (*Answer Relevance*): refere-se à ideia de que a resposta gerada deve abordar a pergunta real fornecida. RAGAS solicita que o LLM gere n perguntas potenciais com base na resposta final. Para cada pergunta q_i, calcula a similaridade $sim(q, q_i)$ com a questão original q, calculando o cosseno entre os embeddings correspondentes. A relevância da resposta, para a questão q é então calculada de acordo com a Equação 5.

$$\text{Answer Relevance} = \frac{1}{n}\sum_{i=1}^{n} sim(q, q_i) \tag{5}$$

Utilização do Contexto (*Context Utilization*): A resposta gerada é dividida em declarações (*claims*) usando um modelo de extração semântica (por exemplo, via LLM ou heurística). Cada declaração é então verificada contra o contexto recuperado usando um modelo de inferência textual (NLI). A métrica é calculada como:

$$\text{Context Utilization@K} = \frac{\sum_{k=1}^{K}(\text{Precision@k} \times v_k)}{\text{Número total de item relevantes nos top-}K\text{ resultados}} \tag{6}$$

$$\text{Precision@k} = \frac{\text{true positives@k}}{\text{true positives@k} + \text{false positives@k}} \tag{7}$$

Onde K é o número total de *chunks* recuperados (contexto) e $v_k \in \{0,1\}$ é o indicador de relevância do rank k. e pode ser julgado por um LLM.

6.5. Conclusões e oportunidades de pesquisa

O RAG na área da saúde apresenta um campo importante para pesquisa, com potencial para transformar a prática clínica, educação médica e pesquisa biomédica [Ng et al. 2025], especialmente por sua capacidade de integrar conhecimento externo confiável ao processo de geração textual. É fundamental que sistemas RAG para aplicações médicas foquem no acesso rápido e preciso a informações atualizadas e relevantes, minimizando os riscos de "alucinações"ou informações equivocadas que podem ser perigosas em um contexto clínico. Existem desafios significativos, mas as oportunidades para impacto positivo são maiores, exigindo colaboração entre pesquisadores de IA, profissionais da saúde e especialistas em informática médica.

Muitos sistemas RAG utilizam mecanismos genéricos de recuperação, como o BM25 ou embeddings treinados em domínios amplos, o que pode levar à seleção de documentos irrelevantes. Para aumentar a precisão e relevância no contexto clínico, é essencial desenvolver estratégias de recuperação especializadas como o fine-tuning de modelos de embeddings usando benchmarks do domínio da saúde no idioma em questão. Explorar técnicas de recuperação que utilizem ontologias médicas para enriquecer a busca pode melhorar buscas mais específicas.

Podem existir casos onde diferentes fontes apresentam recomendações múltiplas e divergentes dependendo do contexto clínico. Um caminho a ser explorado é projetar mecanismos de fusão de contexto que consigam identificar e ponderar múltiplas evidências, integrando critérios como qualidade da fonte, data de publicação e força da evidência.

A utilização de modelos de linguagem via API levanta preocupações quanto à privacidade de dados sensíveis, especialmente em domínios como a saúde. Uma alternativa interessante é a especialização de LLMs não tão grandes, mas com bom poder de resposta, capazes de operar em uma infraestrutura viável para instituições de saúde.

Destacam-se também várias oportunidades de aplicações do RAG no domínio da saúde:

- Sistemas de apoio à decisão clínica como sistemas de diagnóstico assistido que recuperam e resumem informações de guias médicos, literatura recente e prontuários eletrônicos para auxiliar no diagnóstico.

- Sistemas que contemplem alertas clínicos inteligentes que monitoram prontuários eletrônicos e recuperam informações relevantes para alertar sobre interações medicamentosas ou condições emergentes.

- Tutores virtuais baseados em RAG em sistemas que respondam a perguntas de médicos e estudantes recuperando as informações mais atualizadas.

- Melhorias no atendimento ao paciente como chatbots avançados que fornecem informações precisas recuperadas de fontes confiáveis e/ou a geração de explicações personalizadas para pacientes baseadas em seu perfil e nas evidências disponíveis.

Uma das principais preocupações na aplicação de RAG em contextos sensíveis como a saúde: o uso indevido do *self-knowledge*, ou seja, quando o modelo de linguagem utiliza conhecimentos "internos"adquiridos durante o pré-treinamento em vez de se restringir às informações recuperadas. Em tarefas clínicas, esse comportamento pode gerar alucinações factuais perigosas, pois o modelo pode gerar respostas convincentes baseadas em dados genéricos, desatualizados ou errôneos, mesmo quando o conteúdo recuperado não dá suporte àquela afirmação. Controlar o uso do conhecimento paramétrico (*self-knowledge*) dos LLMs em RAG na saúde é importante e demanda uma combinação de engenharia de *prompts*, arquitetura, técnicas de *reranking* e de fusão, métricas de avaliação e treinamento de modelos aptos a não responder como trabalhos vistos na Seção 6.3.4.4. O ideal é que o sistema não só gere respostas baseadas em fatos, mas também demonstre de onde a informação foi tirada.

Referências

[Alsentzer et al. 2019] Alsentzer, E., Murphy, J., Boag, W., Weng, W.-H., Jindi, D., Naumann, T., and McDermott, M. (2019). Publicly available clinical BERT embeddings. In Rumshisky, A., Roberts, K., Bethard, S., and Naumann, T., editors, *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 72–78, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

[Bencke et al. 2024] Bencke, L., Paula, F. S., dos Santos, B. G., and Moreira, V. P. (2024). Can we trust llms as relevance judges? In *Simpósio Brasileiro de Banco de Dados (SBBD)*, pages 600–612. SBC.

[Bogan et al. 2025] Bogan, R., Gaievski, M., Shah, M., and Kolchina, F. (2025). Introducing reciprocal rank fusion for hybrid search.

[Brown et al. 2020] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

[Carchiolo and Malgeri 2025] Carchiolo, V. and Malgeri, M. (2025). Trends, challenges, and applications of large language models in healthcare: A bibliometric and scoping review. *Future Internet*, 17(2):76.

[Chowdhury et al. 2025] Chowdhury, M., He, Y. V., Higham, A., and Lim, E. (2025). Astrid–an automated and scalable triad for the evaluation of rag-based clinical question answering systems. *arXiv preprint arXiv:2501.08208*.

[Cormack et al. 2009] Cormack, G. V., Clarke, C. L., and Buettcher, S. (2009). Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 758–759.

[Cortes et al. 2024] Cortes, E. G., Vieira, R., and Barone, D. A. C. (2024). Perguntas e respostas. In Caseli, H. M. and Nunes, M. G. V., editors, *Processamento de Linguagem Natural: Conceitos, Técnicas e Aplicações em Português*, book chapter 16. BPLN, 2 edition.

[Dettmers et al. 2023] Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. (2023). Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115.

[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 *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.

[Ding et al. 2023] Ding, N., Qin, Y., Yang, G., Wei, F., Yang, Z., Su, Y., Hu, S., Chen, Y., Chan, C.-M., Chen, W., et al. (2023). Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235.

[Duarte et al. 2024a] Duarte, A., Marques, J., Graça, M., Freire, M., Li, L., and Oliveira, A. (2024a). Lumberchunker: Long-form narrative document segmentation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 6473–6486.

[Duarte et al. 2024b] Duarte, A. V., Marques, J. D., Graça, M., Freire, M., Li, L., and Oliveira, A. L. (2024b). LumberChunker: Long-form narrative document segmentation. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N., editors, *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 6473–6486, Miami, Florida, USA. Association for Computational Linguistics.

[Eminagaoglu 2022] Eminagaoglu, M. (2022). A new similarity measure for vector space models in text classification and information retrieval. *Journal of Information Science*, 48(4):463–476.

[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.

[Fan et al. 2018] Fan, A., Lewis, M., and Dauphin, Y. (2018). Hierarchical neural story generation. In Gurevych, I. and Miyao, Y., editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

[Freitag and Al-Onaizan 2017] Freitag, M. and Al-Onaizan, Y. (2017). Beam search strategies for neural machine translation. In Luong, T., Birch, A., Neubig, G., and Finch, A., editors, *Proceedings of the First Workshop on Neural Machine Translation*, pages 56–60, Vancouver. Association for Computational Linguistics.

[Gao et al. 2023] Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, H., and Wang, H. (2023). Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2:1.

[Gu et al. 2021] Gu, Y., Tinn, R., Cheng, H., Lucas, M., Usuyama, N., Liu, X., Naumann, T., Gao, J., and Poon, H. (2021). Domain-specific language model pretraining for biomedical natural language processing. *ACM Transactions on Computing for Healthcare (HEALTH)*, 3(1):1–23.

[Holtzman et al. 2019] Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. (2019). The curious case of neural text degeneration. In *International Conference on Learning Representations*.

[Hu et al. 2022] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. (2022). Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

[Huggingface 2024] Huggingface (2024). Rag worfflow.

[Izacard et al. 2021] Izacard, G., Caron, M., Hosseini, L., Riedel, S., Bojanowski, P., Joulin, A., and Grave, E. (2021). Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

[Järvelin and Kekäläinen 2002] Järvelin, K. and Kekäläinen, J. (2002). Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446.

[Jeong et al. 2024] Jeong, M., Sohn, J., Sung, M., and Kang, J. (2024). Improving medical reasoning through retrieval and self-reflection with retrieval-augmented large language models. *Bioinformatics*, 40(Supplement_1):i119–i129.

[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*. Prentice Hall.

[Kamath et al. 2019] Kamath, U., Liu, J., and Whitaker, J. (2019). *Deep learning for NLP and speech recognition*, volume 84. Springer.

[Kamradt 2024] Kamradt, G. (2024). Semantic chunking. `https://github.com/FullStackRetrieval-com/RetrievalTutorials/tree/main/tutorials/LevelsOfTextSplitting`.

[Ke et al. 2024] Ke, Y., Jin, L., Elangovan, K., Abdullah, H. R., Liu, N., Sia, A. T. H., Soh, C. R., Tung, J. Y. M., Ong, J. C. L., and Ting, D. S. W. (2024). Development and testing of retrieval augmented generation in large language models–a case study report. *arXiv preprint arXiv:2402.01733*.

[Kuriki P. and R. 2024] Kuriki P., Kay F., B. C. and R., P. (2024). Rag worfflow. Disponível em: https://annualmeeting.siim.org/wp-content/uploads/2024/04/4021_Kuriki_RadPointGPT-A-Generative-Chatbot.pdf, último acesso em 21.04.2025.

[Lee et al. 2021] Lee, J., Wettig, A., and Chen, D. (2021). Phrase retrieval learns passage retrieval, too. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3661–3672.

[Lee et al. 2020] Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., and Kang, J. (2020). Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

[Li et al. 2025] Li, X., Ren, W., Qin, W., Wang, L., Zhao, T., and Hong, R. (2025). Analyzing and reducing catastrophic forgetting in parameter efficient tuning. In *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5.

[Li et al. 2023] Li, Y., Li, Z., Zhang, K., Dan, R., Jiang, S., and Zhang, Y. (2023). Chatdoctor: A medical chat model fine-tuned on a large language model meta-ai (llama) using medical domain knowledge. *Cureus*, 15(6).

[Mikolov et al. 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[Moreira 2024] Moreira, V. P. (2024). Recuperação de informação. In Caseli, H. M. and Nunes, M. G. V., editors, *Processamento de Linguagem Natural: Conceitos, Técnicas e Aplicações em Português*, book chapter 21. BPLN, 3 edition.

[Ng et al. 2025] Ng, K. K. Y., Matsuba, I., and Zhang, P. C. (2025). Rag in health care: a novel framework for improving communication and decision-making by addressing llm limitations. *NEJM AI*, 2(1):AIra2400380.

[Orengo and Huyck 2001] Orengo, V. M. and Huyck, C. R. (2001). A stemming algorithmm for the portuguese language. In *spire*, volume 8, pages 186–193.

[Ouyang et al. 2022] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. (2022). Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*.

[Paes et al. 2024] Paes, A., Vianna, D., and Rodrigues, J. (2024). Modelos de linguagem. In Caseli, H. M. and Nunes, M. G. V., editors, *Processamento de Linguagem Natural: Conceitos, Técnicas e Aplicações em Português*, book chapter 17. BPLN, 3 edition.

[Pan et al. 2024] Pan, J. J., Wang, J., and Li, G. (2024). Survey of vector database management systems. *The VLDB Journal*, 33(5):1591–1615.

[Pennington et al. 2014] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

[Porter 1980] Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, 14(3):130–137.

[Qdrant 2025] Qdrant (2025). Fusion vs reranking.

[Rafailov et al. 2023] Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. (2023). Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741.

[Robertson et al. 2009] Robertson, S., Zaragoza, H., et al. (2009). The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

[Ru et al. 2024] Ru, D., Qiu, L., Hu, X., Zhang, T., Shi, P., Chang, S., Jiayang, C., Wang, C., Sun, S., Li, H., et al. (2024). Ragchecker: A fine-grained framework for diagnosing retrieval-augmented generation. *arXiv preprint arXiv:2408.08067*.

[Shinn et al. 2023] Shinn, N., Cassano, F., Berman, E., Gopinath, A., Narasimhan, K., and Yao, S. (2023). Reflexion: Language agents with verbal reinforcement learning.

[Singhal et al. 2023] Singhal, K., Azizi, S., Tu, T., Mahdavi, S. S., Wei, J., Chung, H. W., Scales, N., Tanwani, A., Cole-Lewis, H., Pfohl, S., et al. (2023). Large language models encode clinical knowledge. *Nature*, 620(7972):172–180.

[Singhal et al. 2025] Singhal, K., Tu, T., Gottweis, J., Sayres, R., Wulczyn, E., Amin, M., Hou, L., Clark, K., Pfohl, S. R., Cole-Lewis, H., et al. (2025). Toward expert-level medical question answering with large language models. *Nature Medicine*, pages 1–8.

[Taylor et al. 2022] Taylor, R., Kardas, M., Cucurull, G., Scialom, T., Hartshorn, A., Saravia, E., Poulton, A., Kerkez, V., and Stojnic, R. (2022). Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*.

[Trulens 2024] Trulens (2024). The rag triad.

[Unlu et al. 2024] Unlu, O., Shin, J., Mailly, C. J., Oates, M. F., Tucci, M. R., Varugheese, M., Wagholikar, K., Wang, F., Scirica, B. M., Blood, A. J., et al. (2024). Retrieval-augmented generation–enabled gpt-4 for clinical trial screening. *NEJM AI*, 1(7):AIoa2400181.

[Vaswani et al. 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

[Wang et al. 2022] Wang, L., Yang, N., Huang, X., Jiao, B., Yang, L., Jiang, D., Majumder, R., and Wei, F. (2022). Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

[Wei et al. 2022] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

[Wu et al. 2024] Wu, Z., Hasan, A., Wu, J., Kim, Y., Cheung, J., Zhang, T., and Wu, H. (2024). Knowlab_aimed at mediqa-corr 2024: Chain-of-though (cot) prompting strategies for medical error detection and correction. In *proceedings of the 6th clinical natural language processing workshop*, pages 353–359.

[Xiong et al. 2024] Xiong, G., Jin, Q., Lu, Z., and Zhang, A. (2024). Benchmarking retrieval-augmented generation for medicine. In Ku, L.-W., Martins, A., and Srikumar, V., editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 6233–6251, Bangkok, Thailand. Association for Computational Linguistics.

[Yang et al. 2022] Yang, X., Chen, A., PourNejatian, N., Shin, H. C., Smith, K. E., Parisien, C., Compas, C., Martin, C., Costa, A. B., Flores, M. G., et al. (2022). A large language model for electronic health records. *NPJ digital medicine*, 5(1):194.

[Zakka et al. 2024] Zakka, C., Shad, R., Chaurasia, A., Dalal, A. R., Kim, J. L., Moor, M., Fong, R., Phillips, C., Alexander, K., Ashley, E., et al. (2024). Almanac—retrieval-augmented language models for clinical medicine. *Nejm ai*, 1(2):AIoa2300068.

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

[Zhang et al. 2024] Zhang, T., Patil, S. G., Jain, N., Shen, S., Zaharia, M., Stoica, I., and Gonzalez, J. E. (2024). Raft: Adapting language model to domain specific rag. In *First Conference on Language Modeling*.

[Zhao et al. 2025] Zhao, J., Ji, Z., Fan, Z., Wang, H., Niu, S., Tang, B., Xiong, F., and Li, Z. (2025). Moc: Mixtures of text chunking learners for retrieval-augmented generation system. *arXiv preprint arXiv:2503.09600*.

[Zheng et al. 2023] Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., et al. (2023). Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

